UNIVERSIDAD POLITECNICA DE VALENCIA

E S C U E L A P O L I T E C N I C A S U P E R I O R D E G A N D I A

I.T. Telecomunicación (Sist. Electrónicos)





"Control de velocidad de un motor DC mediante la utilización de un sensor magnético y microcontrolador"

TRABAJO FINAL DE CARRERA

Autor/es:
Joan Escrivà Arlandis

Director/es:
Enrique Colomar Pous
José Pelegrí Sebastiá

GANDIA, 2013

Indice General

1. Introducción

Este proyecto presenta el diseño de un prototipo que permitirá el control total de la transmisión de potencia en un motor DC (corriente continua), de tal manera, que se pueda ajustar la respuesta de aceleración y frenado del motor mediante cuatro parámetros, ajustándose a las necesidades del usuario, que en la aplicación final se utilizará, para regular la respuesta, a través de un sensor magnético, que actuará como acelerador y freno. La transmisión de potencia al motor, se realizará mediante modulación por ancho de pulsos (PWM, siglas en inglés *pulse-width modulation*) utilizando una etapa de potencia controlada por un microcontrolador PIC.

1.1.Visión general

Uno de los juguetes que ha alcanzado gran popularidad en España, ha sido el Scalextric, cuya denominación se ha generalizado utilizándola exclusivamente para dicho juego, independientemente de quien sea el fabricante.

La historia de Scalextric, se remonta al año 1.952 en Inglaterra. Por aquél entonces, existía una pequeña empresa dedicada a la fabricación de juguetes, denominada MINIMODELS. Esta firma, reproducía a escala objetos y utensilios cotidianos, tales como una máquina de escribir, entre otros. Posteriormente, surgió la idea de reproducir en miniatura coches, valiéndose para ello de algunos modelos que entonces competían en las carreras: Maserati, Autounión, etc.

Dichos coches estaban hechos de latón, con gran profusión de detalles. Por razones de marketing, Minimodels decidió poner un nombre propio a ésta gama de coches, al objeto de diferenciarlos del resto de productos que fabricaban. La escala en que se construían los modelos era muy variable, es decir, que cada coche se fabricaba con un tamaño diferente. Por ello, a la gama de coches se le denominó "SCALEX" (Palabra compuesta por "SCALE" y la "X" , que significa "Escala desconocida").

El Sr. Fred Francis, dueño de la empresa Minimodels, ~~y~~ persona emprendedora, tuvo la feliz idea de "motorizar" algunos de los coches. El "motor " era muy curioso, ya que estaba compuesto de un mecanismo de cuerda, que se "cargaba" dándole vueltas a las ruedas traseras. El juguete tuvo tanto éxito que Minimodels decidió probar con motores eléctricos. Estos motores procedían de unos trenes eléctricos que por entonces fabricaba la empresa TRI-ANG. A Partir de ese momento, la gama de coches SCALEX se convirtió en SCALEXTRIC, contracción de las palabras **SCALEX** y ELEC**TRIC**.

Pero la inventiva de Mr. Francis no tenía límite, al poco tiempo, incorporó a los coches una guía con contactos, y diseñó una pista en la que éstos corrían a través de unos carriles que poseían unos contactos a ambos lados. La palabra inglesa de éstos carriles es "SLOT", es entonces cuando aparece la primera pista de Scalextric.

El control de estos motores se realizaba con un mando o controlador (**Figura 1**) que se conectaba con unas bananas o conector (según el fabricante) entre la fuente de alimentación y la pista, constaba de un pulsador que accionaba un reostato, que según la profundidad de este pulsador variaba la resistencia del mismo, y a su vez, el voltaje suministrado a la pista, haciendo la función de divisor de tensión (**Figura 2**).



Figura 1. Mandos comercializados en el año 1962.

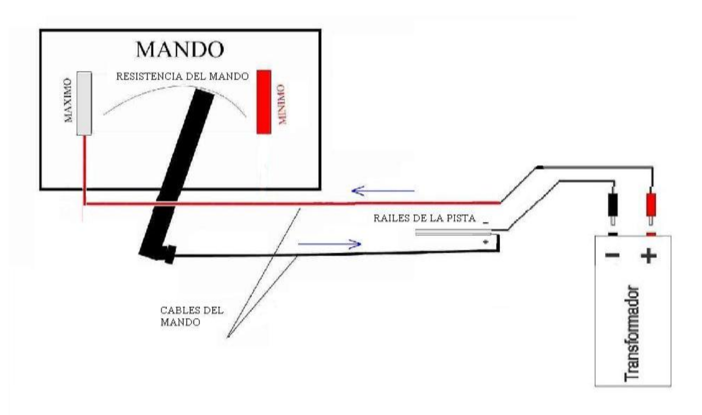


Figura 2. Esquema de conexión del mando sin freno.

Hoy en día, dichos mandos o controladores han evolucionado según las necesidades de los jugadores o pilotos, son más ergonómicos, pues su diseño ha cambiado, adquiriendo forma de "pistola" (**Figura 3**) e incorporando al control la posibilidad de frenado, ya que en los anteriores no existía (**Figura 4**).



Figura 3. Mando comercializado en el año 1993.

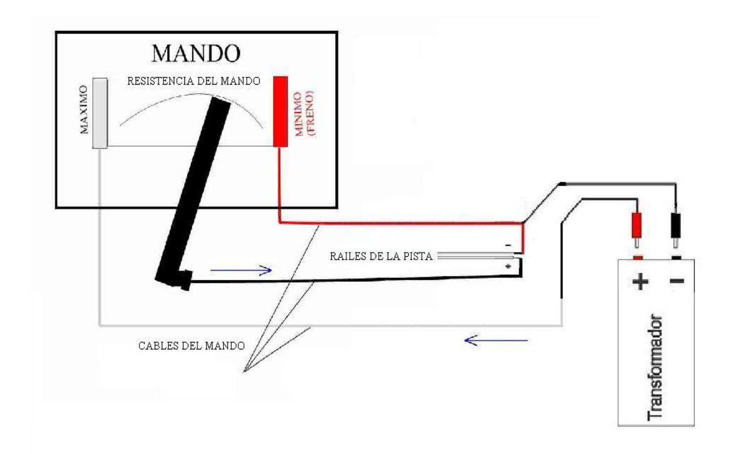


Figura 4. Esquema de conexión del mando con freno.

Con este último control se logra que el motor funcione de tres formas distintas:

1. **Aceleración**: El motor recibe corriente a través del sistema eléctrico de la pista, dicha corriente pasa de los raíles de la misma hasta el motor a través de unas trencillas. Como consecuencia de la transmisión de corriente al motor, éste hace girar el eje del mismo, que a su vez, hace mover al coche a través del conjunto mecánico, formado por: el piñón, la corona, el eje, las ruedas,...

2. **Desaceleración**: Que consiste en la carencia de suministro de corriente al motor, o bien, en un menor suministro de corriente al ocasionado en la aceleración, en cualquiera de los dos casos, debido a la inercia del conjunto coche-motor, el vehículo tarda en desacelerar o detenerse. Sin embargo, mientras el coche se desplaza sin corriente, o con menos corriente de la necesaria, según el caso, el motor deja de funcionar como tal, pues al continuar girando por las ruedas pasa a funcionar como generador, ya que se suministra menos corriente al motor de la que necesita para alcanzar su velocidad deseada, por lo cual genera corriente en vez de ser absorbida. Todo ello hace, que aparezca una corriente en la pista generada por dicho motor que ahora funciona como generador, la corriente generada, se disipará cuando el motor quede girando a la velocidad correspondiente con la presión a la que se está presionando el gatillo, pasando un espacio de tiempo, durante el cual, el sistema vuelve al equilibrio, será entonces, cuando el motor funcionará como tal, dejando de hacerlo como generador.

3. **Frenado**: El frenado consiste en consumir esa energía generada por el motor durante la desaceleración lo más rápidamente posible, ya que cuanto mayor sea la demanda de energía eléctrica por el generador, mayor será la inercia necesaria para que éste, siga en movimiento. Como la inercia es aproximadamente la misma, siempre que desaceleremos, si aumentamos la demanda de energía eléctrica conseguiremos que el coche se detenga antes, pues obligamos al "generador" a consumir más movimiento para mantener esa energía eléctrica. La forma de incrementar la demanda de energía eléctrica, será provocando un cortocircuito temporal en la pista.

Este cortocircuito temporal, se puede controlar por medio de una resistencia variable, de esta forma, se conseguirá un cortocircuito de menos intensidad, todo ello implica, que la energía consumida en el cortocircuito sea menor, y por tanto, también es menor el consumo de energía cinética, es decir, menos freno.

A partir de estos tipos de funcionamiento, los diferentes fabricantes del este sector han desarrollado mandos electrónicos, dichos mandos controlan el motor mediante modulación por ancho de pulsos (**PWM**) a través del microprocesador, llevan ajustes de curva de aceleración y frenado modificables con opción de guardado en memoria (**Figura 5**).



Figura 5. Mando digital comercializado en la actualidad.

1.2.Motivación

Como usuario de este juego o *hobby* se ha visto la necesidad de aumentar las prestaciones de los productos que ya hay comercializados, incorporando configuraciones nuevas, o incluso modificando o sustituyendo componentes, por nuevas tecnologías alternativas capaces de aumentar la precisión y calidad del producto, sin encarecerlo.
Esto se debe a que cada mando o controlador que hay en el mercado, presenta unas configuraciones o prestaciones diferentes entre sí, y que no comparten con el resto de productos similares, lo que los hace singulares, debido a sus prestaciones y precio, siendo para los usuarios una difícil elección, por lo que conlleva a una posible pérdida de dinero e interés sobre este juego o *hobby*, en el caso de no hacer una buena elección.
Por tanto, el presente proyecto se centrará en agrupar los mejores ajustes, componentes y configuraciones que se pueden realizar en esta aplicación, obteniendo un producto completo con las mejores tecnologías que existen hoy en día en el mercado, ofreciendo calidad y facilidad de uso de este producto al mismo coste.

1.3.Objetivos

En este proyecto tal como se ha comentado en el apartado anterior, se centrará en la recopilación de las configuraciones esenciales para el control óptimo del motor, y a su vez del coche, con los componentes más avanzados que existen hoy en día y con una configuración fácil de ajustar, incluyendo la regulación del freno, aceleración y un *antispin* (control de tracción).
Dichas configuraciones, se realizarán a través de encoders rotatorios, que permite digitalmente un ajuste de las mismas, más preciso, los encoders tienen la ventaja respecto a los potenciómetros de no sufrir tanto desgaste mecánico, influyendo a la larga en el ajuste, también tienen la ventaja que mientras el dispositivo esté apagado, estos controles no afectan a la configuración interna del mando, evitando así, un desajuste involuntario.
Como no se va a utilizar dichos potenciómetros, se incorporará una interfaz visual **LCD** (*Liquid Crystal Display*), que permitirá tener todos los valores de ajuste de las configuraciones mencionadas anteriormente en pantalla, con ello, obtenemos una innovación, ya que ningún mando o controlador existente en el mercado lleva esta pantalla.
Otra innovación más para este tipo de aplicación, es el control del gatillo acelerador, que funciona con un sensor magnético **GMR** (*Giant Magneto Resitance*) que comparado con los mandos que hay en el mercado, este, es el único que funciona con sensor **GMR**, utilizando los demás, sensores **Hall** o codificación óptica, perdiendo resolución y precisión respecto al **GMR**.
También se ha hecho una extracción de la etapa de potencia, capaz de controlar el motor de la placa principal acoplándolo a una caja externa, permitiendo el ahorro de espacio en el interior del mando, peso y evitando sobrecalentamientos por falta de refrigeración, soportando así grandes corrientes y evitando la ruptura de componentes causadas por la temperatura ocasionada.
Todo ello, se controlará mediante un microprocesador **PIC 18F4520,** que permitirá guardar hasta cincuenta configuraciones diferentes, ya que ningún mando hasta el momento, lo ha permitido.

1.4.Organización de los contenidos

En el punto 1, se ha realizado una breve explicación de la historia, evolución y funcionamiento en lo que se refiere a la aplicación final del presente proyecto, a su vez se han expuesto las características básicas de la misma, así como las ventajas de la tecnología empleada respecto a lo existente en el mercado.

En el punto 2, se presenta el prototipo del circuito que se divide en cinco partes:

- Conexión de *encoders* y funcionamiento.
- *Interface* visual **LCD** y su conexión.
- Sensor magnético **GMR** del gatillo y acondicionamiento.
- Etapa de potencia **PWM.**
- Conexión ICSP del microprocesador **PIC** y ensamblaje del prototipo.

En el punto 3, se describen las características y las unidades funcionales del microprocesador **PIC 18F4520,** así como el diagrama de flujo que se utilizará para implementar el código en C para el correcto funcionamiento de todas las partes descritas en el punto 2.

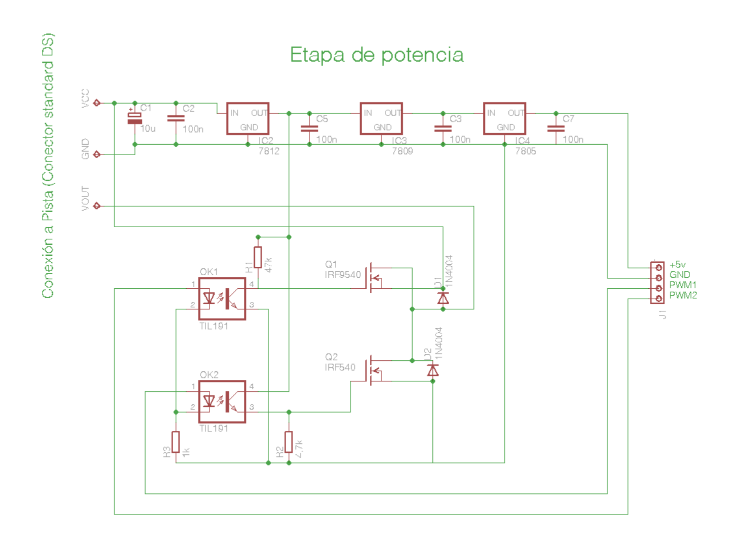
En el punto 4, se detalla el manual de usuario, así como la configuración del prototipo.

En el punto 5, se relaciona toda la bibliografía utilizada para la realización del presente proyecto.

El punto 6, se adjunta todos los *datasheets* de los componentes más importantes utilizados en el prototipo, programa en C y el esquema de conexiones de todo el montaje como el *layout*.

2.Esquema del circuito impreso del prototipo

Para poder realizar el circuito impreso del prototipo, se ha diseñado el esquemático con todos los componentes que se van a necesitar y conexiones entre ellos (**Figura 6**).
Para ello, se ha trabajado con el programa de diseño de esquemáticos y *Layouts* denominado **EAGLE PCB DESIGN v6.1** puesto que es fácil de utilizar, rápido y posee una amplia librería de componentes.



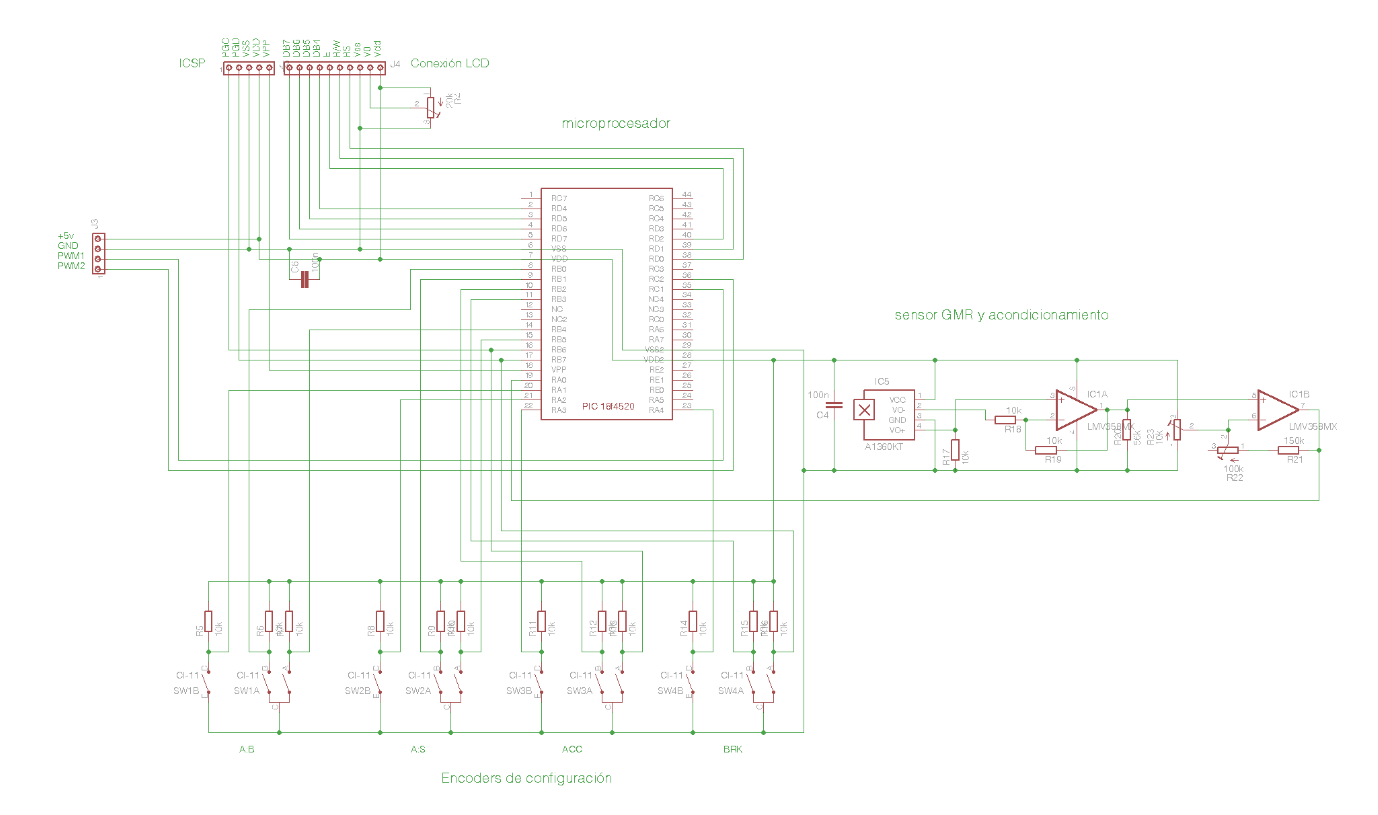


Figura 6. Esquema de conexiones del prototipo completo.

2.1.Conexión de *encoders* y su funcionamiento

En este prototipo se ha optado por la utilización de *encoders* rotativos, ya que dichos componentes son muy simples de conectar e implementar en el PIC, respecto a los potenciómetros convencionales. Una de las ventajas de utilizar *encoders*, consiste en que en nuestra aplicación se utilizan cuatro controles, que en el supuesto de emplear los potenciómetros, se tendrían que utilizar conversores A/D para obtener el valor de cada potenciómetro retardando en cada conversión el programa, en cambio con los *encoders*, solamente hacen falta dos entradas digitales, una para codificar el sentido de rotación, y la otra para el pulsador (**Figura 7**).

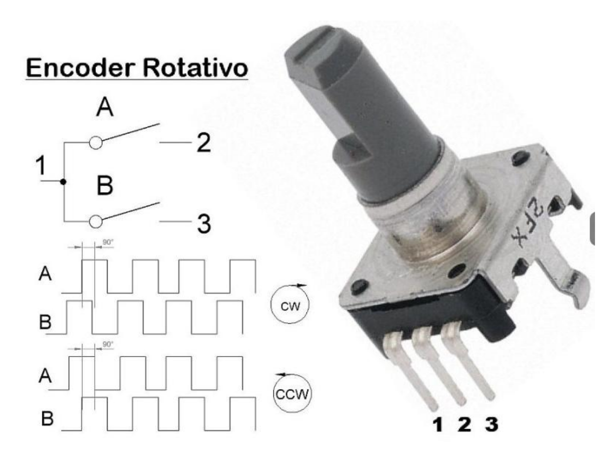


Figura 7. Ilustración del encoder rotativo y su funcionamiento.

En este caso, como se van a utilizar cuatro controles, se conectarán los cuatro *encoders* en el Puerto B del PIC, para la detección de la rotación se conectará el pin 2 de cada *encoder* a la parte baja del Puerto B (RB0,RB1,RB2,RB3) y el pin 3 a la parte alta respectivamente (RB4,RB5,RB6,RB7).
Utilizando la interrupción del Puerto B en el PIC se puede detectar que el *encoder* se ha utilizado en cada momento, así como la dirección de giro.
En el circuito se conectará el pin 1 de *cada encoder* a 5V y los pines 2 y 3 a una resistencia de 10 KΩ a GND para generar las señales en cuanto se accione (**Figura 8**).

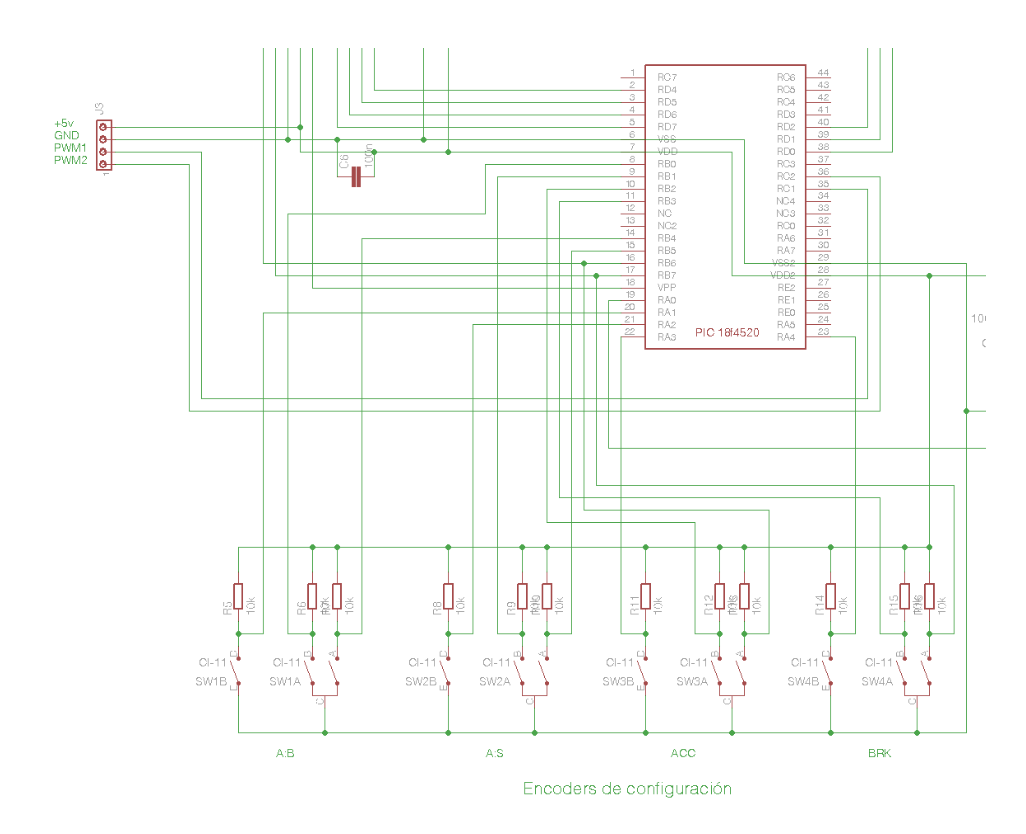


Figura 8. Esquema de conexión de los encoders al microprocesador.

2.2. *Interface* visual **LCD** y su conexión

La utilización de una pantalla LCD en este prototipo, es para que haga la función de interface visual entre el prototipo y el usuario, este LCD no solo mostrará el valor de cada *encoder* que configurara el usuario en todo momento, también le ayudará a navegar por el menú de configuración, con el fin de poder guardar o cargar las configuraciones nuevas o ya guardadas anteriormente en la memoria.

El LCD que se ha utilizado es de 2 x 16 caracteres alfanuméricos con retro iluminación en color amarillo (**Figura 9**), la tensión de alimentación es de + 5V con entrada **V0** entre 0 y +5V. Para regular el contraste de los caracteres, se puede configurar el bus de datos para que pueda trabajar con 4 u 8 bits, en este caso, se ha configurado a 4 bits, con el fin de ahorrar un puerto en el microcontrolador.

La conexión entre el LCD y el microcontrolador se ha realizado mediante un conector de pines, para que se pueda desmontar la pantalla y poderla ensamblar en el prototipo, siguiendo la tabla de conexiones de cada pin, que nos suministra el fabricante del LCD, (**Figura 10**) se han conectado los pines necesarios al Puerto D del microcontrolador (**Figura 11**).



Figura 9. Ilustración del display LCD.

LCD			Microprocesador
Pin	Símbolo	Función	Puerto D
1	Vss	GND	
2	Vdd	5V	
3	V0	Contraste	
4	RS	Selección de Registro	RD0
5	R/W	Leer/Escribir datos	RD1
6	E	Señal de Habilitación	RD2
11	DB4	Linia bus de datos	RD4
12	DB5	Linia bus de datos	RD5
13	DB6	Linia bus de datos	RD6
14	DB7	Linia bus de datos	RD7

Figura 10. Tabla de pines de interconexión entre el LCD y el microprocesador.

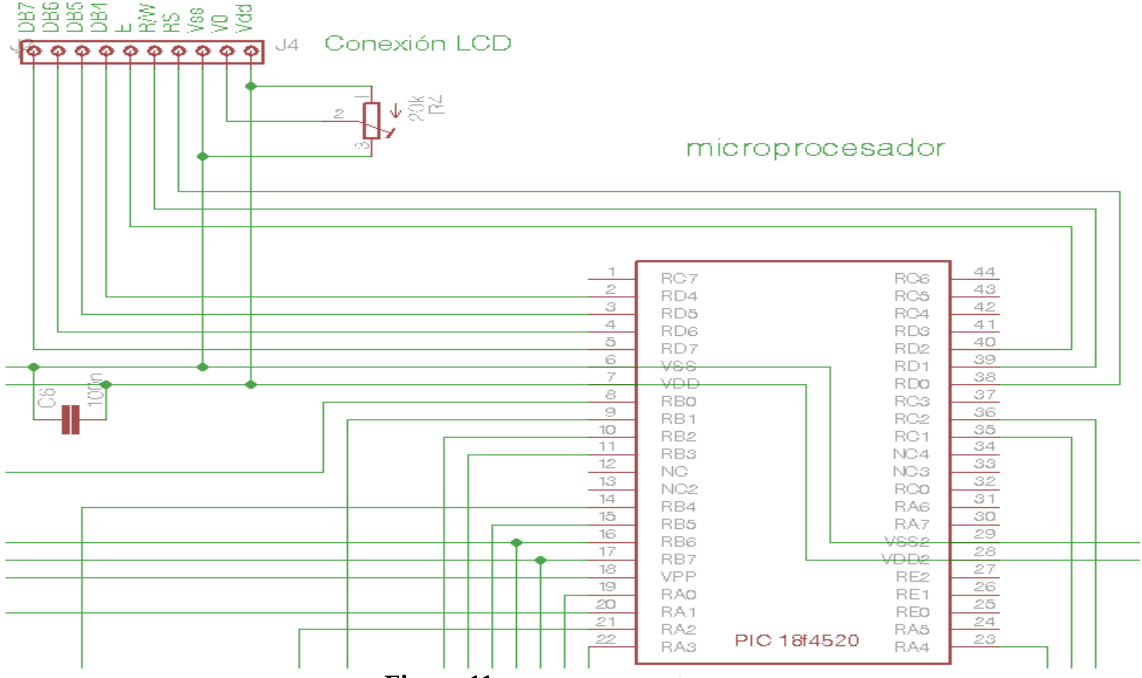


Figura 11. Esquema de conexión del LCD y el microcontrolador.

2.3. Sensor **GMR** del gatillo y su acondicionamiento

Esta parte del proyecto es muy importante e imprescindible para el funcionamiento de todo sistema, pues el prototipo que se ha diseñado, funciona en torno a un gatillo que será el que determine la cantidad de potencia que el usuario elegirá accionando desde el punto de reposo (freno) hasta el tope de recorrido del gatillo (máxima potencia del motor).
Para saber en todo momento la posición del gatillo, se ha elegido un sensor GMR (*Giant Magnetoresistance Effect*). Este, tiene un efecto mecánico cuántico, que se observa en estructuras de película delgada compuestas de capas alternadas ferromagnéticas y no magnéticas.
Se manifiesta en forma de una bajada significativa de la resistencia eléctrica, observada bajo la aplicación de un campo magnético externo: cuando el campo es nulo, las dos capas ferromagnéticas adyacentes tienen una magnetización antiparalela, ya que están sometidas a un acoplamiento ferromagnético débil entre las capas. Bajo el efecto de un campo magnético externo, las magnetizaciones respectivas de las dos capas se alinean, cayendo la resistencia de la multicapa de manera súbita. Los spines de los electrones de la sustancia no magnética, se alinean en igual número, de manera paralela y antiparalela al campo magnético aplicado, y por tanto, sufren un cambio de difusión magnética en una menor medida respecto a las capas ferromagnéticas, que se magnetizan de forma paralela (**Figura 12**).

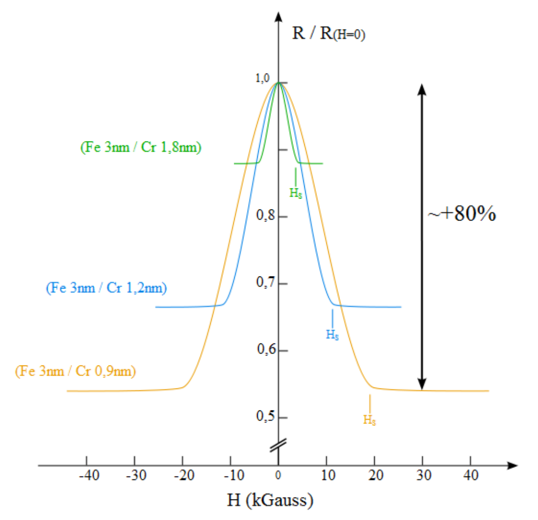


Figura 12. Gràfica del GMR en función del material y campo magnético.

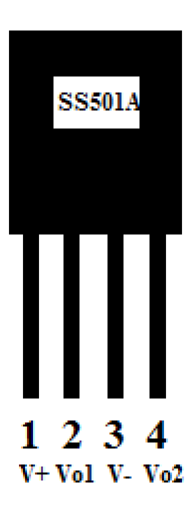
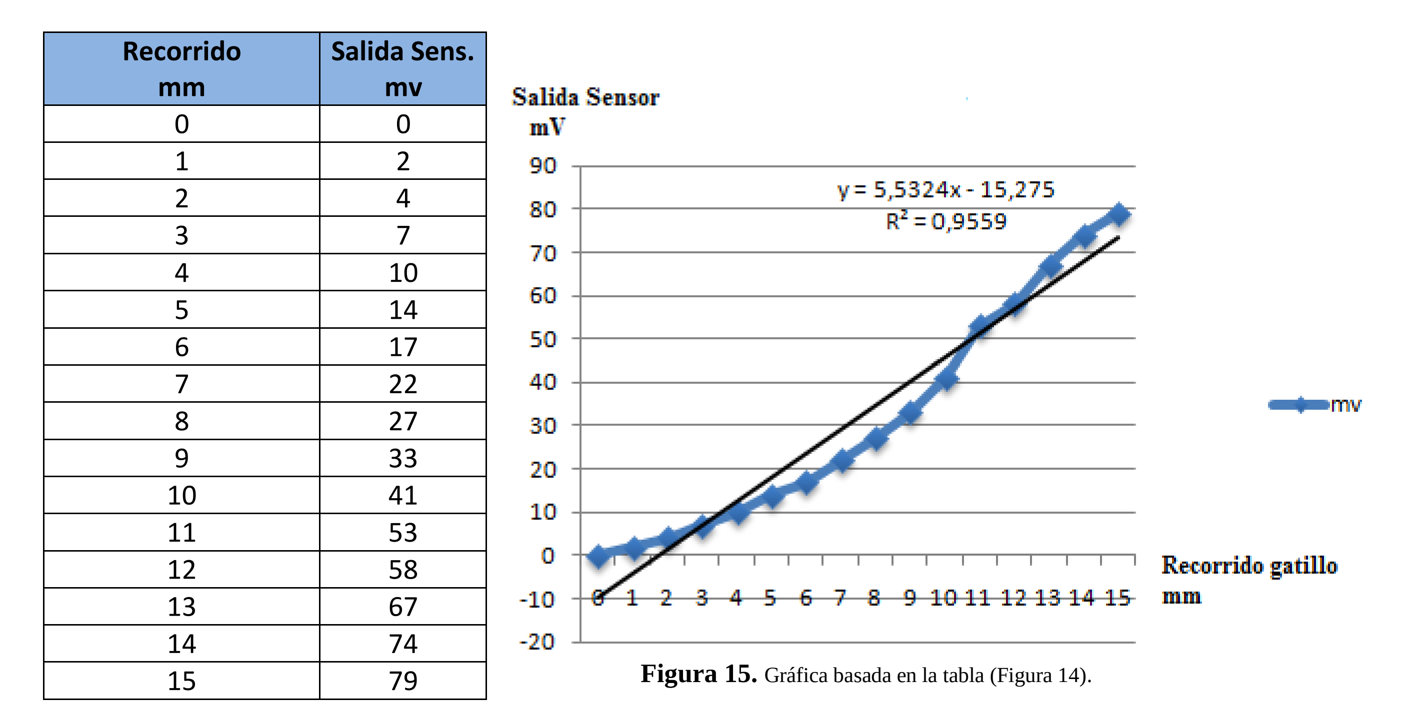


Figura 13. Ilustración del GMR y sus conexiones

En este prototipo, se ha utilizado un sensor GMR experimental que aún no se ha comercializado, y por lo tanto, no tenemos un *datasheet* que indique con exactitud, todos los parámetros y características del mismo.
De la información que hay disponible, se ha obtenido el patillaje del componente para su conexión, sabiendo que la patilla 1, corresponde a la alimentación positiva, mientras que la 3, es la negativa, las patillas 2 y 4, corresponden a la salida de tensión diferencial del sensor (**Figura 13**). Conociendo todo ello, se han realizado mediciones, incrustando un imán de Neodimio en el extremo opuesto del gatillo y situando el sensor al punto, de máximo recorrido del gatillo, de esta forma, se han recogido las mediciones desde la posición del gatillo en el punto de reposo, hasta el tope máximo haciendo una lectura a cada milímetro de recorrido, alimentando el sensor a +5V, obteniéndose una curva de respuesta a la que se ha extraído la función de transferencia (**Figura 14** y **Figura 15**).



Recorrido mm	Salida Sens. mv
0	0
1	2
2	4
3	7
4	10
5	14
6	17
7	22
8	27
9	33
10	41
11	53
12	58
13	67
14	74
15	79

Figura 15. Gráfica basada en la tabla (Figura 14).

Figura 14. Tabla de mediciones con el sensor.

Una vez obtenida la curva de la tensión de salida del sensor GMR se ha procedido a diseñar un circuito acondicionador para la señal diferencial, que este entrega, dicha señal, tiene que ser apta para que el convertidor A/D, incorporado en el PIC, pueda convertir en su totalidad el rango de tensión que el sensor entrega, para ello, dicha señal tiene que ser referida a masa y de un rango desde +0,062V hasta +4,93V (**Figura 16**).

El circuito acondicionador consta de dos amplificadores operacionales *rail to rail*, esto significa, que los rangos mínimo y máximo de la tensión de salida de ambos amplificadores operacionales, sea prácticamente la misma que la tensión de su alimentación, Dichos amplificadores, están integrados en un mismo Circuito Integrado, concretamente se ha utilizado el **LMV358SG-13,** este Circuito Integrado tiene una tensión de alimentación de hasta +5,5V por lo que facilita el montaje del circuito de acondicionamiento, ya que el sensor, así como todo el resto del sistema, trabaja a una tensión de +5V.

El primer amplificador operacional del circuito acondicionador, está configurado como amplificador restador, su función será convertir la señal diferencial del sensor GMR a una tensión referida a masa, para ello, el valor de las resistencias que compone el circuito del primer amplificador operacional, son del mismo valor (R17=R18=R19=10KΩ) así la función de transferencia es:

$$\mathbf{Vo1= (V+ - V-)*1.}$$

El segundo amplificador operacional, también está configurado para que funcione como amplificador restador, la diferencia al anterior, es que este, sí que amplificará la señal entregada por el primer amplificador operacional, para que sea de los niveles de tensión, siendo aptas para la entrada del convertidor A/D, para ello, se utilizará un potenciómetro (R23=10KΩ), que trabajará
como divisor de tensión, entre 0 y +5V, que irá conectada a la entrada negativa del operacional, corrigiendo el *offset* de la señal, si fuera necesario. Una vez montado el proyecto, también se ha introducido otro potenciómetro en la red de realimentación (R22=100KΩ) en serie, con una resistencia (R21=150KΩ) para poder regular la ganancia del amplificador operacional, si fuera necesario una vez montado el proyecto, siendo la función de transferencia:

$$\mathbf{Vo = ((V+-V-)-(5*Ro/10K\Omega))*(1+((150K\Omega + R22) / Ro)).}$$

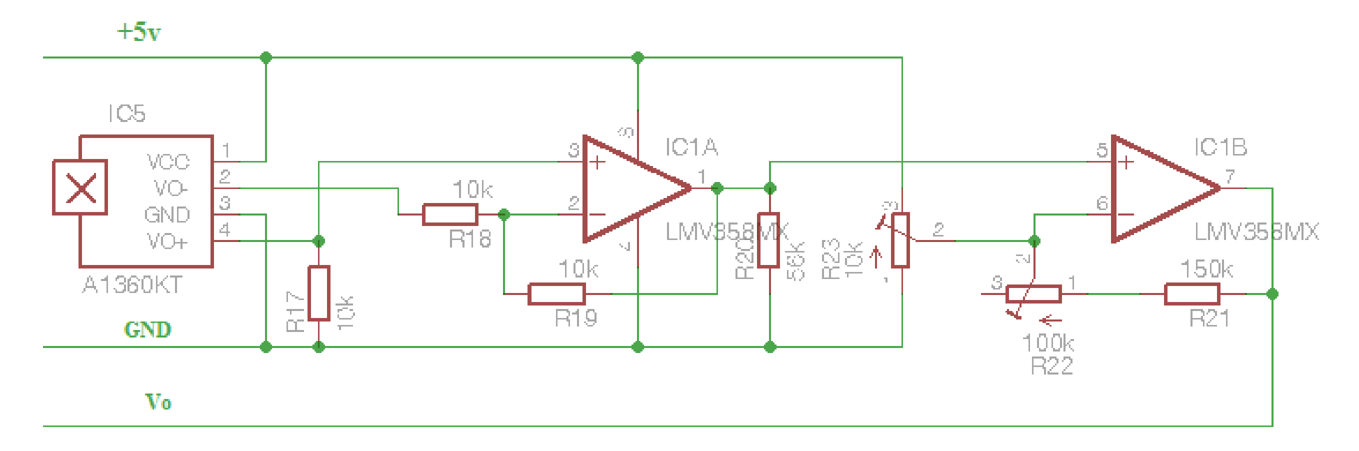


Figura 16. Esquema de conexión del sensor GMR y su acondicionamiento.

2.4. Etapa de potencia **PWM**

Para poder controlar la potencia destinada al motor DC, se va a utilizar la modulación por ancho de pulsos conocida como **PWM** (*pulse-with modulation*), el **PWM** de una señal o fuente de energía, consiste en una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal, o una cuadrada como en este caso), ya sea para transmitir información a través de un canal de comunicaciones, o bien, para controlar la cantidad de energía que se envía a una carga (motor DC como en este caso**)(Figura 17**).

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

Siendo:
D: el ciclo de trabajo.
τ: el tiempo en que la función es positiva (ancho del pulso).
T: el período de la función.

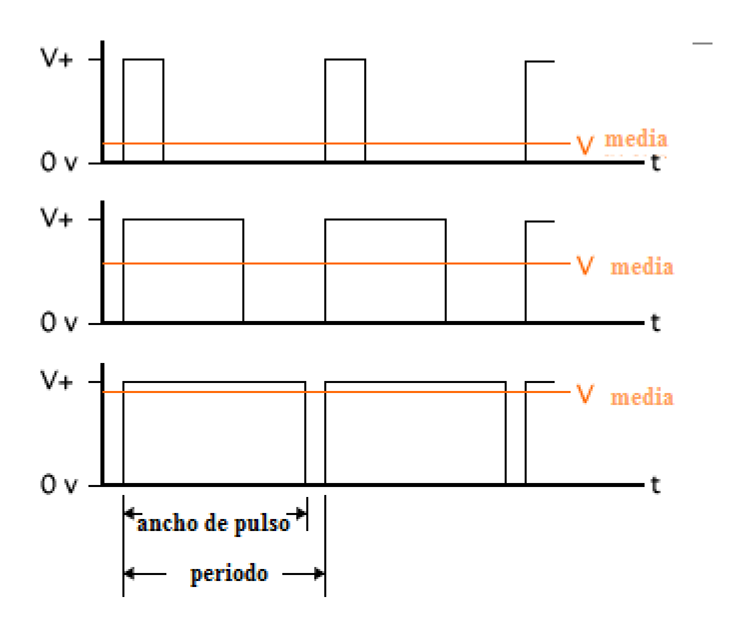


Figura 17. Gráfica del funcionamiento del PWM según el ancho de pulso.

Para el presente prototipo, es necesario la utilización de dos señales **PWM** distintas, una señal, para el control de potencia destinada al motor en un solo sentido de giro, y la otra, para poder controlar la cantidad de potencia en el freno del motor, este freno del motor, se realizará mediante un cortocircuito controlado en los bornes del mismo, provocando un consumo inmediato de la corriente generada por la bobina del motor, cuando este está rodando por la energía cinética que genera el mismo movimiento del coche cuando está rodando por inercia, este consumo inmediato de la corriente generada por el mismo motor, provoca que la bobina del mismo se pare, y como consecuencia, todo el conjunto mecánico del coche se pare.
Una de las unidades funcionales que tiene el microprocesador que se va a utilizar (*Timer 2*), tiene la posibilidad de generar estas dos señales **PWM** a una frecuencia de 300Hz independientemente una de la otra, teniendo disponibles en el microcontrolador la patilla RC1para la señal PWM1 y RC2 para la señal PWM2 respectivamente.

Como la amplitud máxima en voltaje de las señales **PWM** generadas por el microcontrolador no supera los +5V, debiendo trabajar el motor DC a la tensión suministrada al sistema completo por el usuario, se ha diseñado una etapa de potencia, la cual adaptará las señales **PWM** generadas por el microcontrolador mediante dos optoacopladores **TIL191,** que gobernarán un semipuente en H con un transistor NPN **IRF540,** y con otro PNP **IRF9540,** actuando directamente sobre el motor a través del conector **DS**, también regulará la tensión suministrada por el usuario, ajustándola al valor de tensión que se necesite en cada parte del prototipo, a través de tres reguladores de tensión en cascada (**7812 - 7809 - 7805**). De esta forma se reparte la potencia consumida por los reguladores disipando mejor el calor (**Figura 18**).
Debido a la cantidad de calor disipada, espacio y el peso total del conjunto se ha optado por separar esta etapa del sistema completo, uniéndola a la etapa de control a través de un cable de 4 líneas, en la que se le suministrará una tensión referida a masa de +5V para la alimentación de todo el sistema, con las otras dos líneas la etapa de control nos enviará las dos señales **PWM** correspondientes al freno y al acelerador (PWM1 y PWM2).

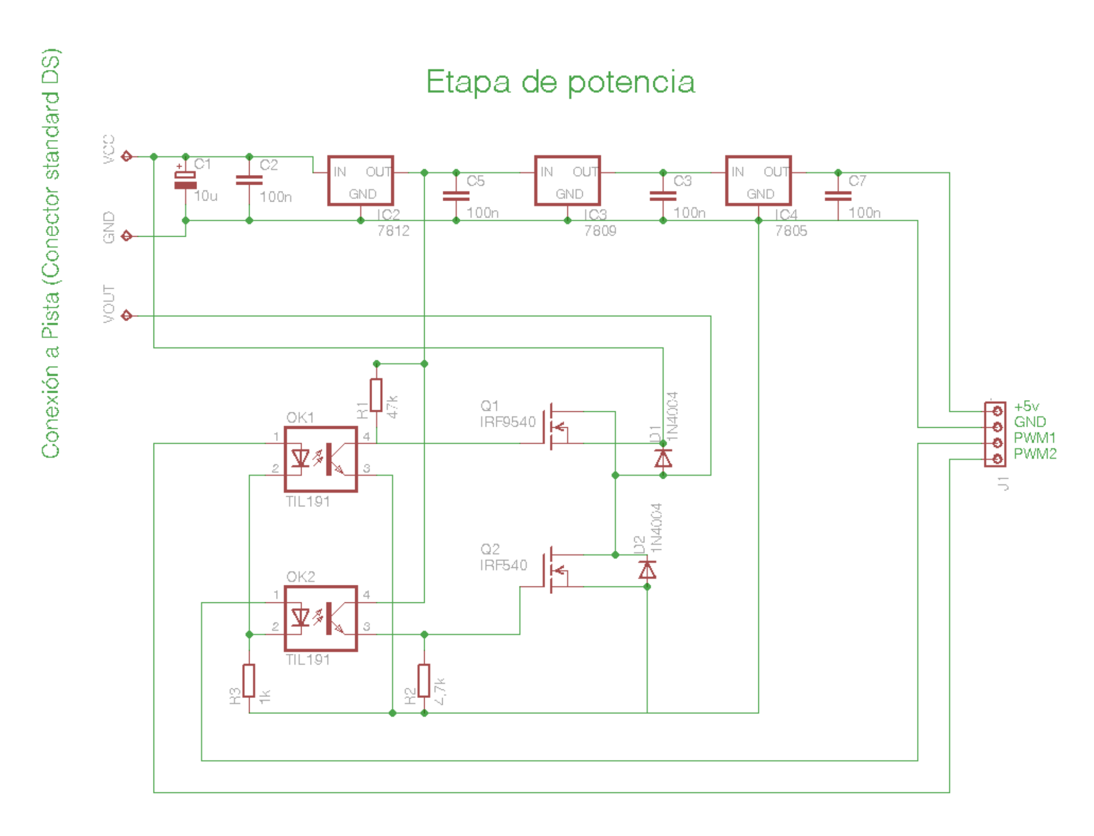


Figura 18. Esquema de conexión de la etapa de potencia.

2.5. Conexión **ICSP** del microprocesador **PIC 18F4520**

Como la gran mayoría de microcontroladores PIC de la marca *Microchip*, tienen incorporados un puerto ICSP (In Circuit Serial Programming), por tanto, una vez que el prototipo se ha soldado y montado en la placa, se podrá programar a través de un bus de cinco líneas, para ello, se ha necesitado un *interface* de comunicación entre el PC y el microcontrolador como el **PICkit 2,** que se ha utilizado en este proyecto (**Figura 19 y 20**).



Figura 19. Programador USB PICkit 2.



Figura 20. Diagrama de flujo de la programación del PIC.

Para el PIC que se ha empleado en este proyecto (**19F4520**), se ha preparado la conexión de las señales del programador **PICkit 2,** mediante pines, que se han soldado a la misma placa del prototipo yendo directamente al microcontrolador, según una tabla que se ha creado a partir del *datasheet* del propio microcontrolador (**Figura 21 y Figura 22**).

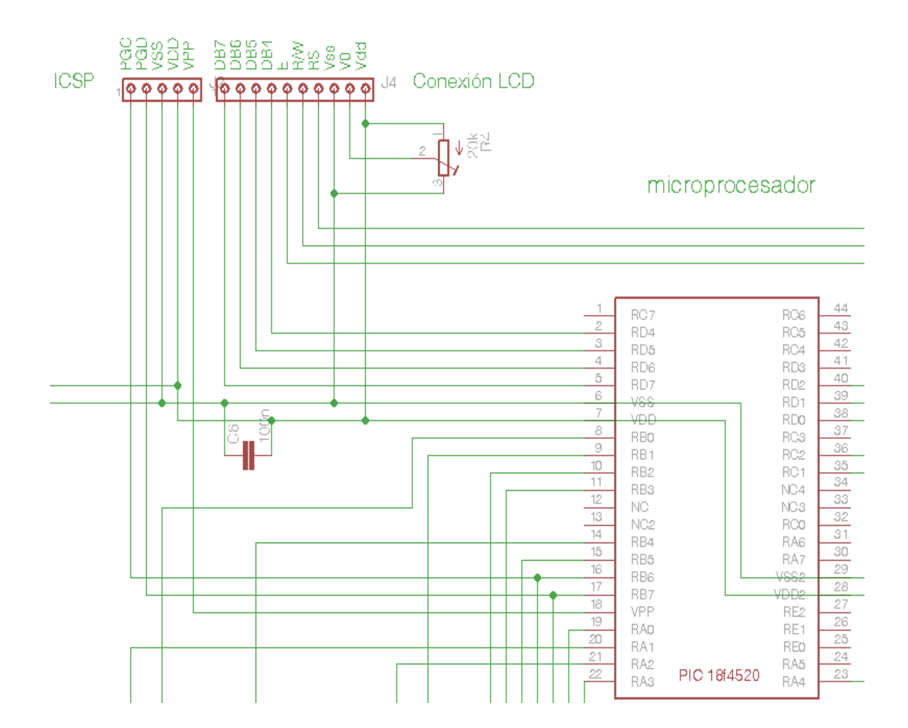


Figura 21. Esquema de conexión entre el microprocesador y los pines de programación.

Pickit 2		PIC 18F4521	
Pines	Función	Pines	Función
1	Vpp	18	Vpp
2	Vdd	7,28	Vdd
3	Vss	6,29	Vss
4	PGD	17	RB7
5	PGC	16	RB6

Figura 22. Tabla de pines para la programación

2.6. Evolución del diseño del prototipo

Teniendo en cuenta todas las partes que van a componer este prototipo se decidió montar y conectar todas ellas en una placa de entrenamiento para empezar con la programación y hacer pruebas con la etapa de potencia y los *encoders* rotatorios (**Figura 23**).

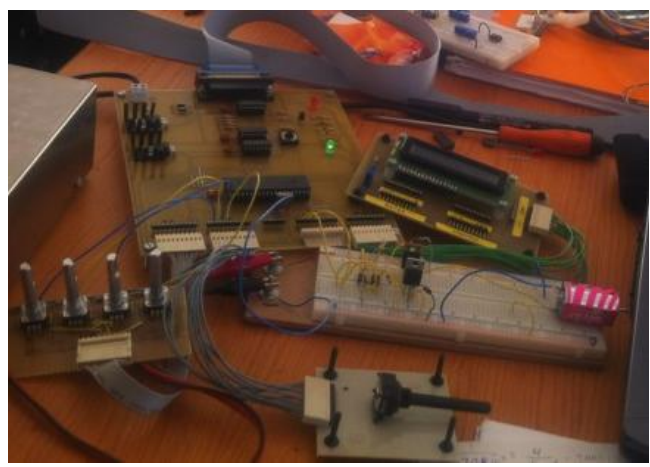
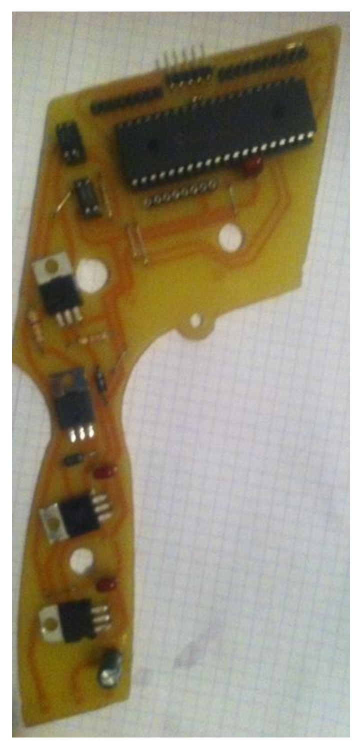


Figura 23. Ilustración de la placa de entrenamiento con PIC 16F877 y periféricos en fase de prueba.

El programa en principio, se diseñó para el microcontrolador que lleva integrada la placa de entrenamiento (PIC16F877) pero se llegó a un punto que por el número de variables y el peso del programa en si este microcontrolador era insuficiente para este proyecto, por lo cual se eligió el microprocesador 18F4520 que además de tener todas las funciones del anterior microcontrolador tiene una memoria de programa con mayor capacidad y un reloj interno configurable a frecuencias más elevadas que las que se trabajaba en el anterior.



A consecuencia del cambio del microcontrolador se diseñó una placa impresa para seguir con la programación del prototipo, el diseño de esta placa se realizó con la forma y medidas para poderla incrustar y ajustar a la carcasa de plástico donde se tiene que montar y para que en la placa final ya sea de las medidas precisas para el ensamblaje, a todo ello situando gran cantidad de los componentes que se sabía con seguridad que se iban a montar para distribuir el espacio que se va a utilizar para el diseño de la placa impresa final (**Figura 24**).

Una vez montada esta placa se adaptó el programa para el funcionamiento en este microcontrolador y se observó que la etapa de potencia alcanzaba unas temperaturas no deseadas ya que estaba situada en la empuñadura del mando aumentando aún más la temperatura de estos componentes, entre esto y el poco espacio que se disponía para montar todos los componentes que quedaban pendientes para su montaje, y se decidió cambiar todos los componentes de inserción a SMD de la etapa de control y separar la etapa de potencia del conjunto del mando, albergándola en un módulo aparte con disipadores, conectándola después con un cable de cuatro hilos creando así el prototipo final.

Figura 24. Ilustración placa de entrenamiento con PIC 18F4520.

2.7. Circuito impreso del prototipo

Una vez realizado el esquemático de todo el prototipo se ha procedido a diseñar el *layout* de la placa impresa con el programa **Eagle PCB Design v6.1.**
La placa se ha diseñado con las dimensiones y forma precisas, para poder introducirla exactamente dentro de una carcasa de plástico comercial adecuada para este propósito (mando de Slot)
(**Figura 25 ,26 y 27**).
Los componentes que se han soldado en la etapa de control, son la gran mayoría SMD, por lo tanto, las dimensiones de las pistas, así como las anchuras de estas, son de un tamaño que la fabricación de la placa impresa obliga a la utilización de fresadora CNC (Control Numérico por Computadora) al igual que el contorno y perforaciones para los componentes, para ello, una vez finalizado el *layout* de la placa impresa, se han extraído los ficheros GERBER de las coordenadas de las perforaciones y fresado de las pistas, así como del contorno de la placa impresa (**Figura 28**).

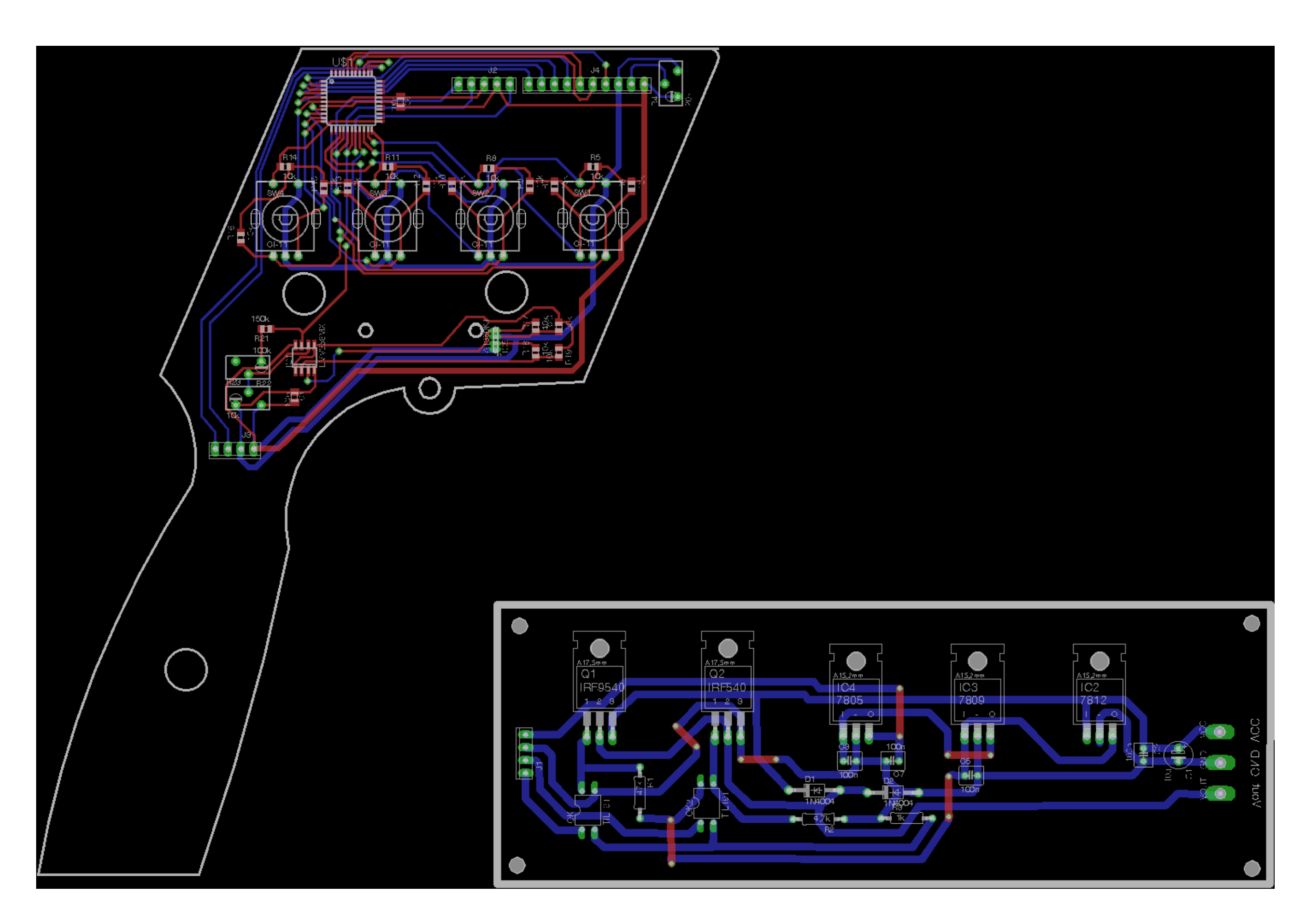


Figura 25. Ilustración del *layout* final de la etapa de control y de potencia.

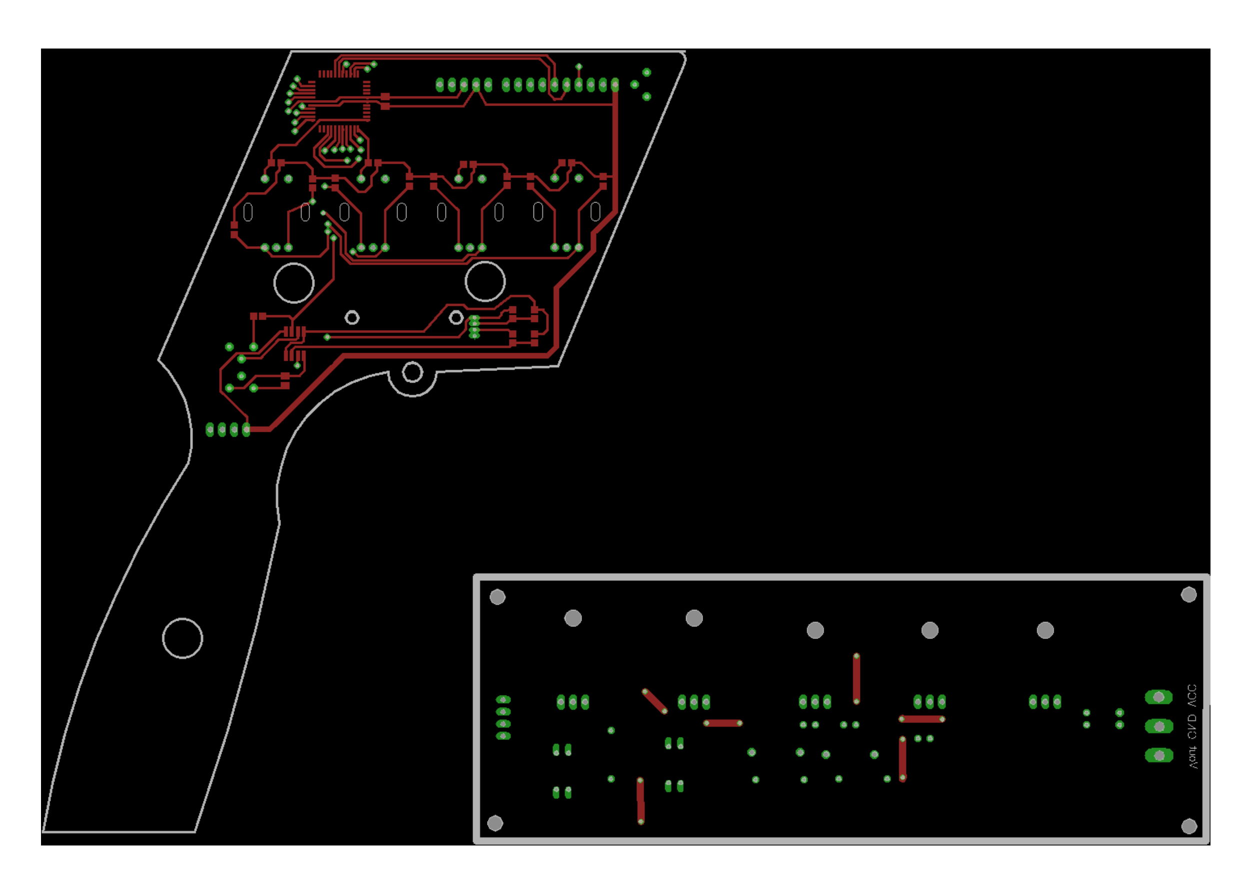


Figura 26. Ilustración del *layout* final vista desde la cara *top.*

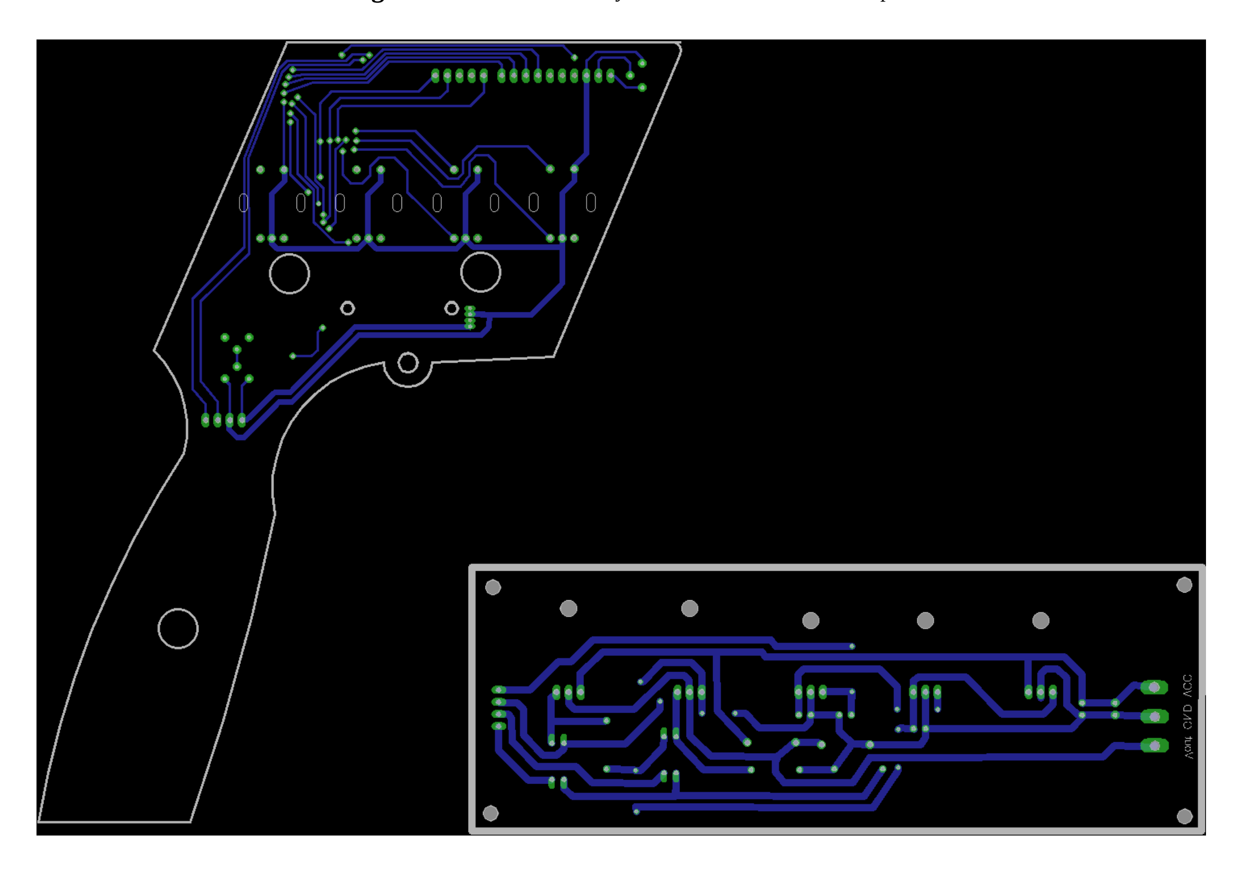


Figura 27. Ilustración del *layout* final vista desde la cara *bottom.*

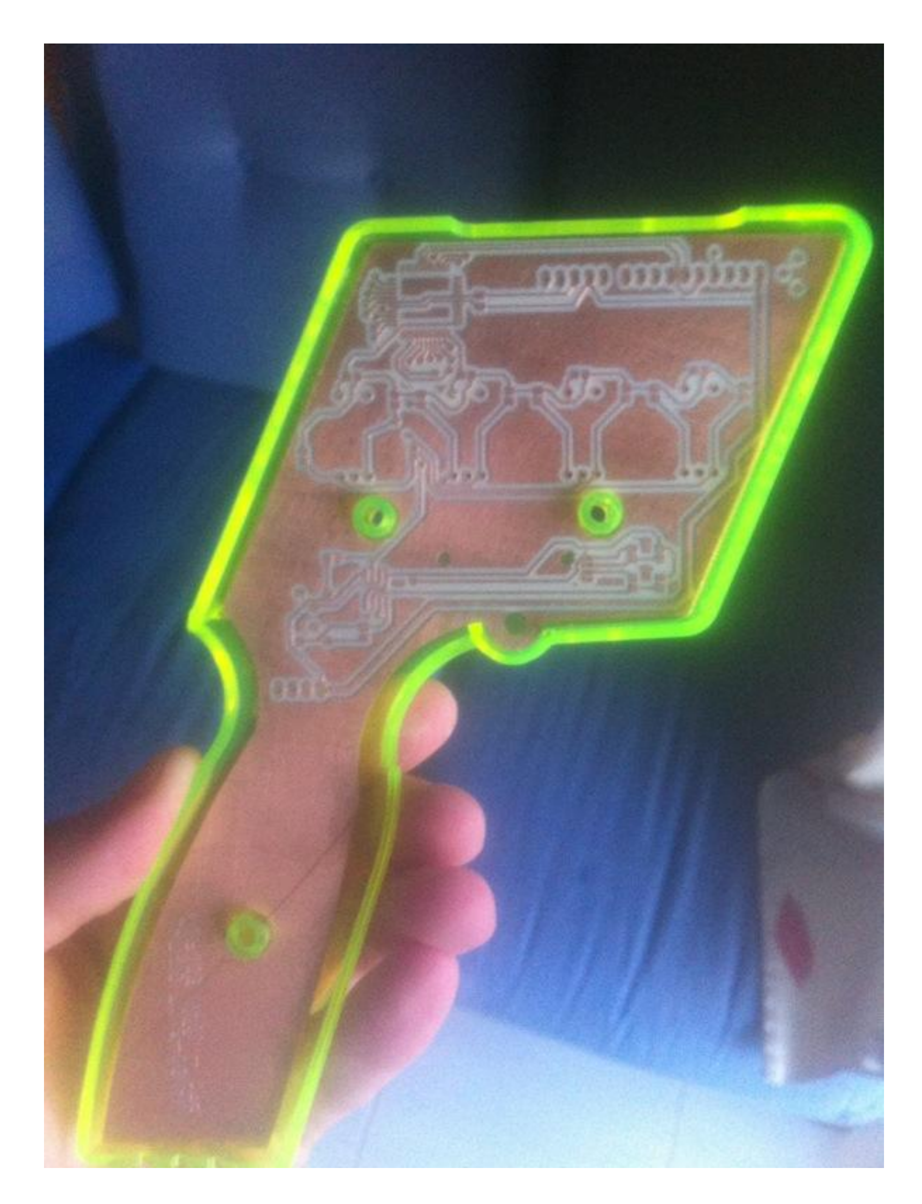


Figura 28. Ilustración de la placa impresa final fresada.

2.8. Ensamblaje del prototipo acabado

Por último, para poder programar y probar el prototipo, una vez se han tenido todos los componentes de las dos etapas soldados, se montarán en la placa impresa de la etapa de control sobre la carcasa de plástico, que es de la marca **PROFESSOR MOTOR** al igual que el gatillo de nylon, al que se le incrustará el imán de neodimio para la detección de posición del sensor **GMR**.
A esta carcasa se le han perforado cuatro agujeros, con el fin de sacar los *encoders* y revestirlos con un tapón de plástico de diferentes colores para poder diferenciar las distintas funciones que realizan los encoders, también se le ha recortado parte de la tapa de plástico de la carcasa, para incrustarle la pantalla LCD.
La etapa de potencia se ha cerrado con una caja de plástico comercial con el fin de proteger la placa impresa, a continuación se han conectado las dos etapas entre sí, mediante un cable Ethernet UTP cat. 5 y el conector DS, con los tres pines de conexión a la pista, ~~que irá~~ soldado con tres cables a la placa impresa de la etapa de potencia (**Figura 29**).

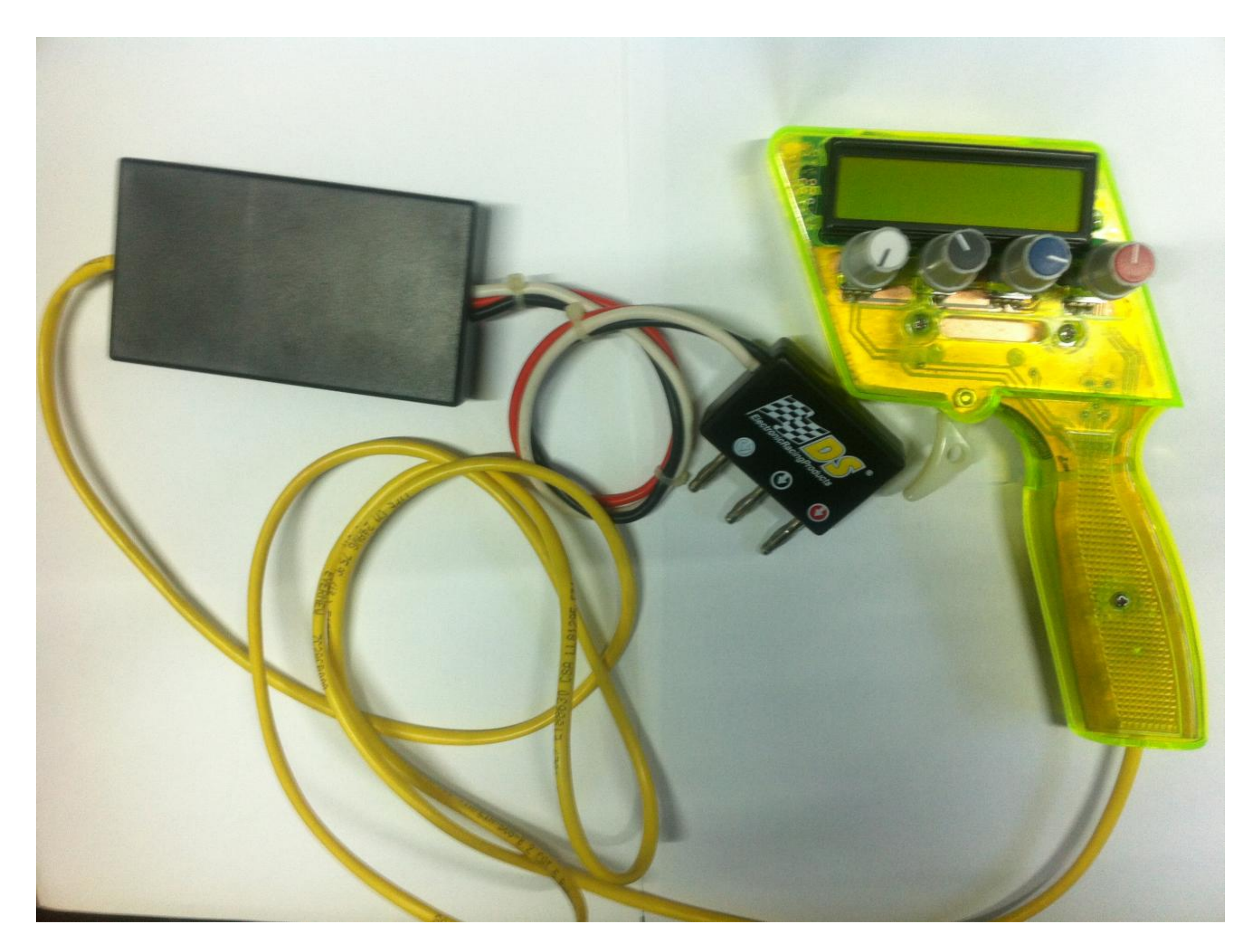


Figura 29. Ilustración del prototipo final ensamblado.

3. Microprocesador PIC 18F4520 y su programación

En este punto del proyecto ya finalizado y testeado todo el circuito y conexiones del prototipo, se han programado todas las funciones que son requeridas en el mismo, y que están integradas en el PIC, para ello, se utilizará el programa **MPLAB IDE v 8.87,** que es un compilador para programación en C, siendo también ensamblador, a parte, se ha instalado un *pluggin* en el programa **MPLAB,** al objeto de poder compilar, optimizando el lenguaje C, para la familia de los PIC 18F llamado **HI-TEC PICC 18,** una vez tenemos el programa compilado y exportado en formato **.hex**, se ha instalado el programa **PICkit 2 Programer v2.5,** con el fin de poder utilizar el PICkit 2, grabando el programa en la memoria del microprocesador (**Figura 30**).

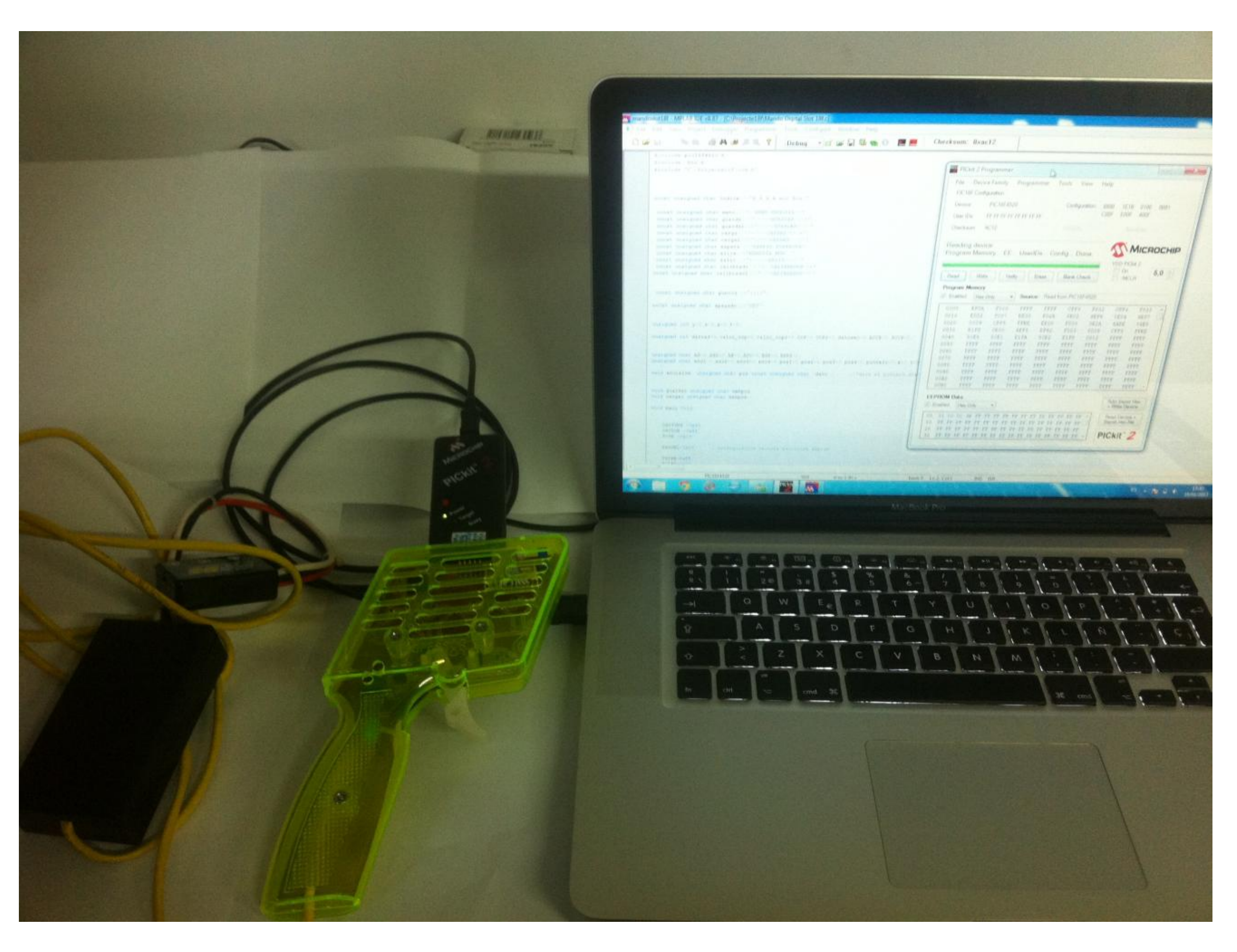


Figura 70. Ilustración de la programación del microcontrolador con *Pickit 2 Programer*.

3.1. Unidades funcionales utilizadas en el **PIC 18F4520**

Para este proyecto se ha utilizado el 90% de las unidades funcionales que nos indica el fabricante que tiene integrado el microcontrolador empleado (**PIC 18F4520**).
De todas las unidades funcionales e interrupciones del programa que integra dicho microcontrolador, se han utilizado:

- Reloj interno, configurado para que trabaje el microcontrolador a 4MHz

- Interrupciones del Puerto B desde RB4 hasta RB7, configurando todo el puerto B, como entradas para conectar los cuatro *encoders*, que sirve para la detección del sentido de giro de estos *encoders*, para ello, han hecho falta dos entradas para cada uno, una entrada para la interrupción y la otra para la entrada digital. Por ejem. (*encoder* nº1: la salida A del *encoder* se conectará a la entrada RB0 y la salida B se conectará a la entrada RB4).

- Funciones de lectura/escritura de la memoria EEPROM, utilizada para guardar y cargar hasta 50 de las distintas configuraciones que el usuario podrá emplear además de la utilizada en ese momento, que aunque se desconecte el sistema sin guardar dicha configuración no se borraría, para ello, en el programa principal se ha programado un contador, para que a una frecuencia de 1 Hz, el programa guarde en la memoria EEPROM uno de los cuatro valores que contiene la configuración elegida por el usuario, esta frecuencia tan baja es debido a que el tiempo de escritura, es más alto que el de lectura, en el supuesto que se realizaran más escrituras por segundo, provocaría una ralentización del programa principal, por lo que a cada cuatro segundos, se guardaría toda la configuración del usuario.

- Convertidor A/D de 10 bits, incorporado en el microcontrolador actúa sobre el puerto A del mismo, configurando el puerto A como entradas digitales, excepto RA0, que utiliza el canal 0 del convertidor como entrada analógica, cuya frecuencia de conversión, se ha programado con la interrupción del Timer 1.

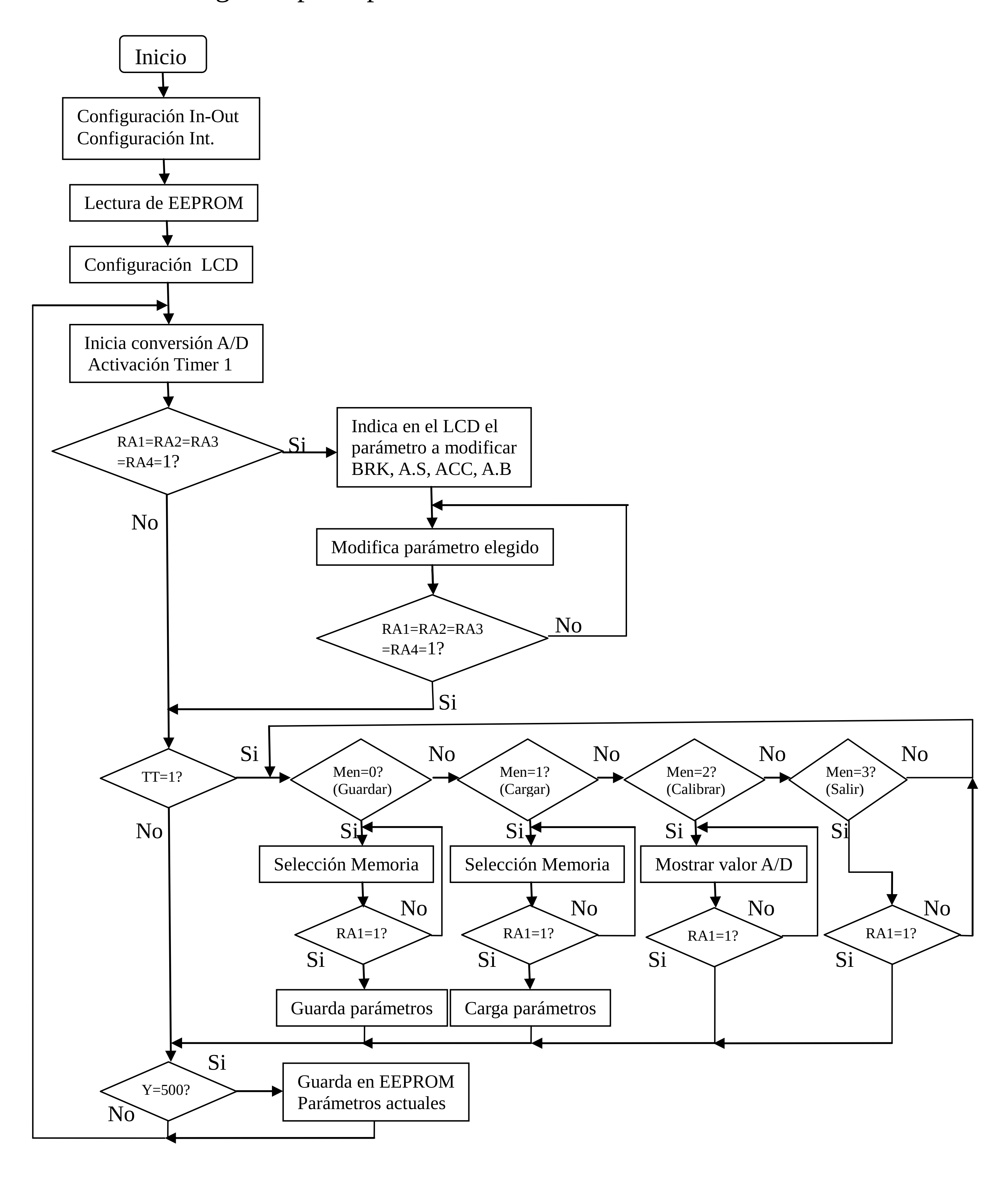
- Timer 0, este temporizador se ha programado con el fin de detectar la pulsación continuada del pulsador del primer *encoder* durante cinco segundos, entrando en el menú de guardado y carga de las configuraciones del usuario.

- Timer 1, dicho temporizador estará programado para que produzca interrupciones a una frecuencia de 100Hz, estas interrupciones, sirven para activar el conversor A/D, recogiendo el dato capturado, para la manipulación y asignación del dato a introducir al ancho de pulso del **PWM**.

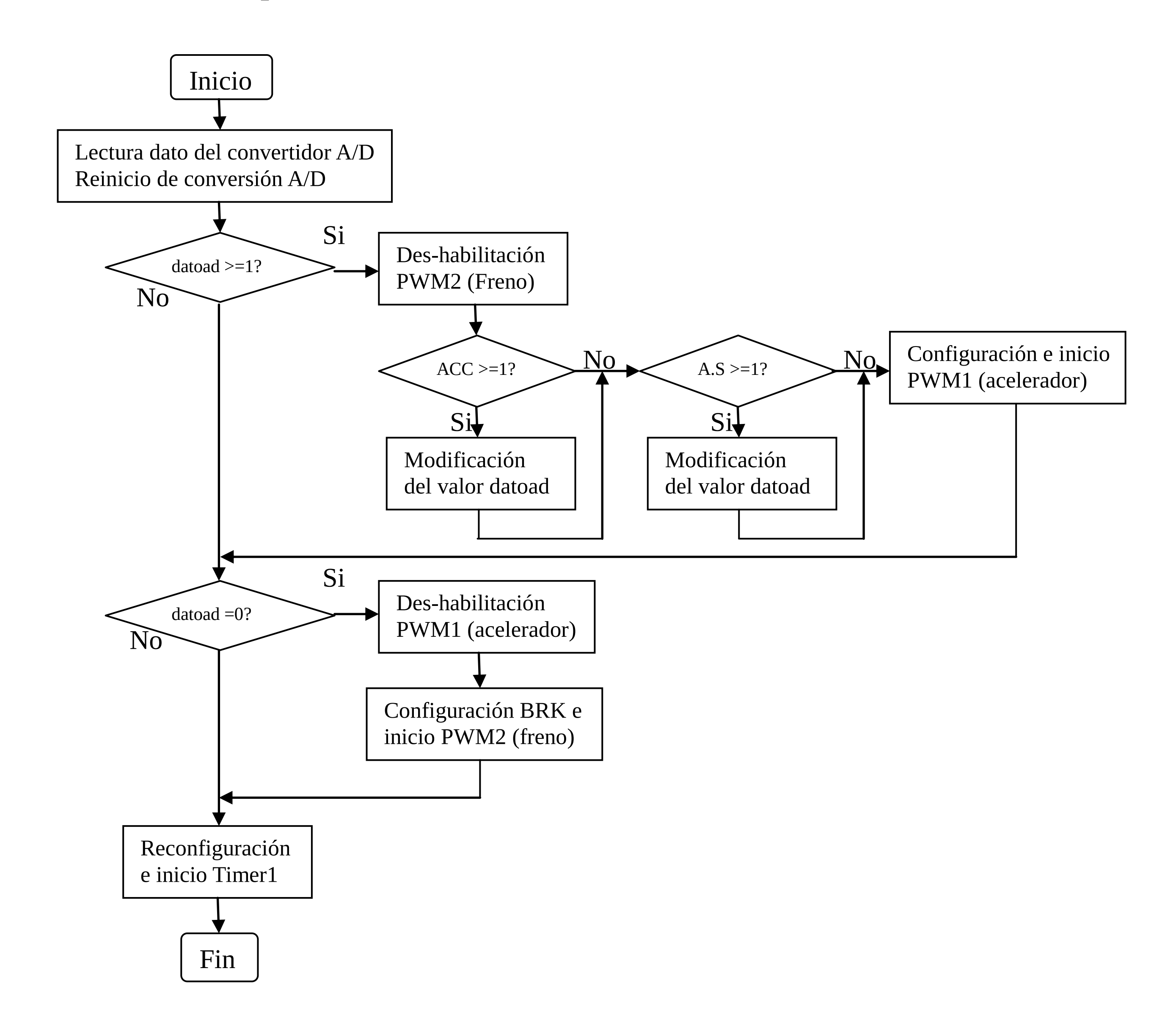
- Timer 2, este timer tiene la ventaja de poder configurarse como temporizador, comparador y **PWM,** en este caso, se ha configurado como **PWM,** teniendo dos salidas independientes RC1 y RC2, siendo una, para el freno, y la otra, para el acelerador, este **PWM** se ha configurado para trabajar a una frecuencia de 300Hz, teniendo una resolución de 10 bits de ciclo de trabajo, facilitando la programación, ya que se puede manipular el valor de ancho de pulso directamente desde el valor capturado en el convertidor A/D.

3.2. Diagrama de flujo del programa

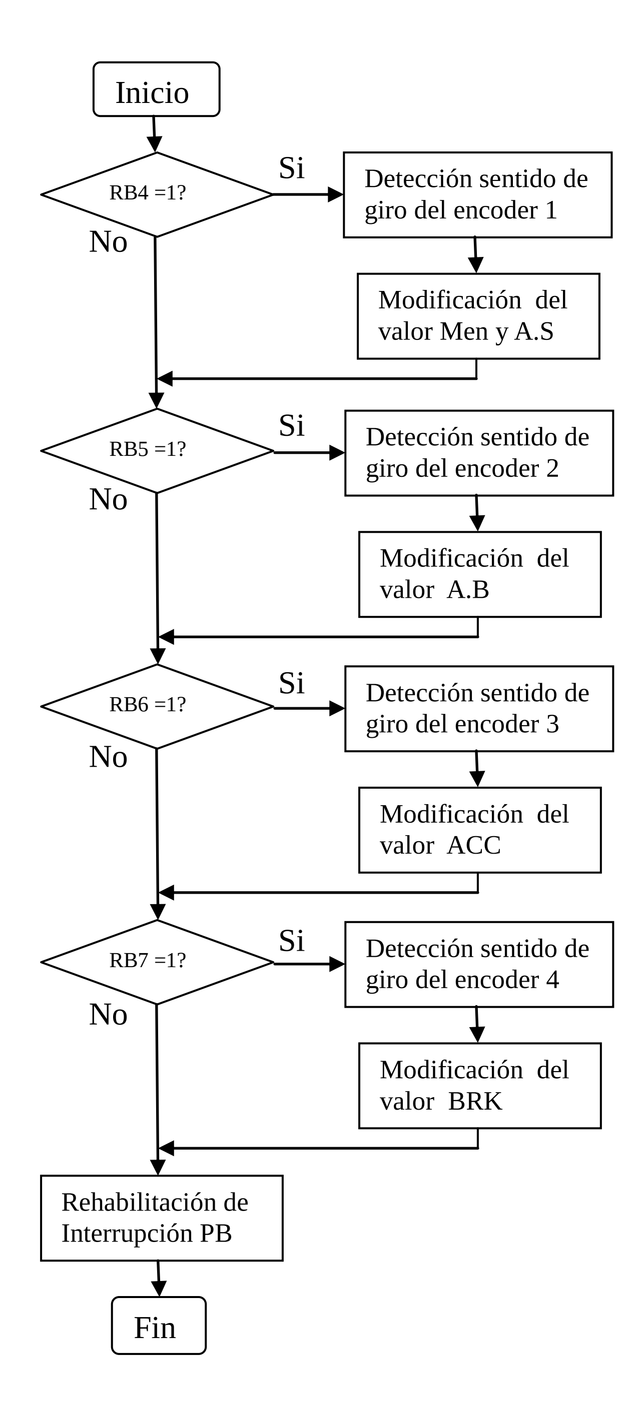
- Programa principal



- Interrupción Timer 1



- Interrupción Puerto B



4.Manual de usuario y configuración del prototipo

Para que el prototipo funcione correctamente y no sufra ningún desperfecto se debe alimentar a una tensión mínima de +8V y una máxima de +20V referida a masa en corriente continua.
Este prototipo equipa con el conector estándar DS de tres pines, solo se tendrá que conectar a la caja de conexiones “stop & go”, del mismo fabricante (DS) (**Figura 31**).



Figura 31. Ilustración de la conexión del prototipo a la caja de conexiones DS.

En el momento en que la caja de conexiones “stop & go” active el suministro de corriente a la pista, se encenderá la retroiluminación del LCD, mostrando los nombres de los parámetros de configuración en orden a los *encoders* en la primera línea, y en la segunda línea muestra el valor anterior al apagado del prototipo de cada parámetro de configuración, indicando que el prototipo ya está en funcionamiento (**Figura 32**).



Figura 32. Ilustración del prototipo en funcionamiento.

El sistema de control de aceleración y freno son muy sencillos, pues se deberá apretar el gatillo para acelerar, y soltarlo para desacelerar o frenar (gatillo en punto de reposo) (**Figura 33 y 34**).

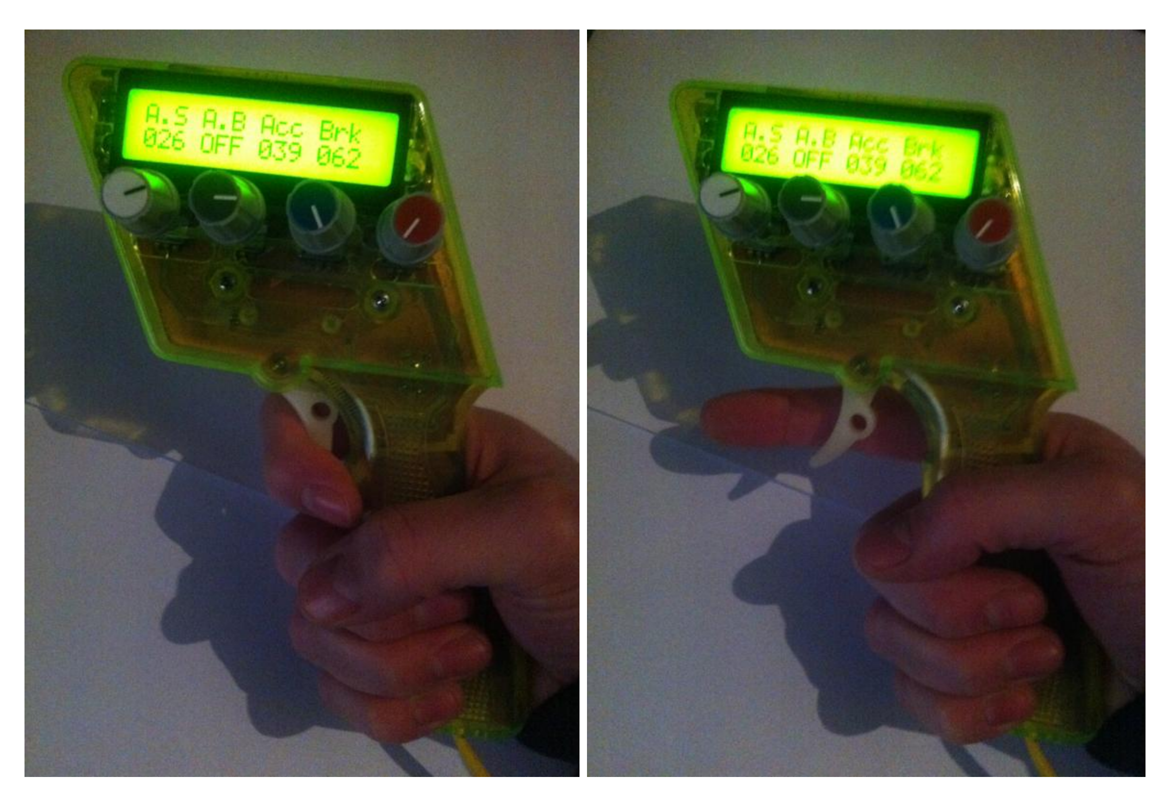


Figura 33. Ilustración de la acción acelerador. **Figura 34.** Ilustración de la acción freno.

Las abreviaturas de los parámetros de configuración que nos indica el LCD ayudan a recordar el significado y parámetro a modificar sobre el *encoder* que se está accionando:

- **A.S.** – *AntiSpin* (Antiderrape): este parámetro sirve para retardar la entrega de potencia al motor, evitando un derrape en pista, pues, a mayor número de A.S mayor retardo de potencia.

- **A.B.** – *Automatic Brake* (Freno automático): dicho parámetro no ha sido implementado en el programa, debido al nuevo reglamento de la ACS del 2013, en el que se prohíbe la utilización de mandos con freno automático.

- **ACC.-** *Acceleration* (Aceleración): en este parámetro se puede aumentar la respuesta de la entrega de potencia al motor, disminuyendo el recorrido del gatillo, ya que a mayor número de ACC, menor será el recorrido, y mayor será la respuesta en potencia.

- **BRK.-** *Brake* (Freno): este parámetro es fundamental, ya que sirve para regular la cantidad de freno que aplica al motor cuando el gatillo está en reposo, pues a mayor número de BRK, mayor será el freno motor.

Para poder modificar el valor de los parámetros de configuración, se tiene que pulsar el *encoder* que indique el parámetro a modificar, en el que aparecerá un puntero que indicará el parámetro seleccionado, que mediante un giro a la derecha se subirá el valor del parámetro, mientras que con un giro a la izquierda lo bajará, el rango de los valores de los cuatro parámetros de configuración oscila entre el 0 (aparecerá en el LCD como **OFF**) y el 100, una vez aparezca en pantalla el valor deseado, se volverá a pulsar el *encoder* para que este quede fijo, desapareciendo el puntero en el LCD (**Figura 35**).

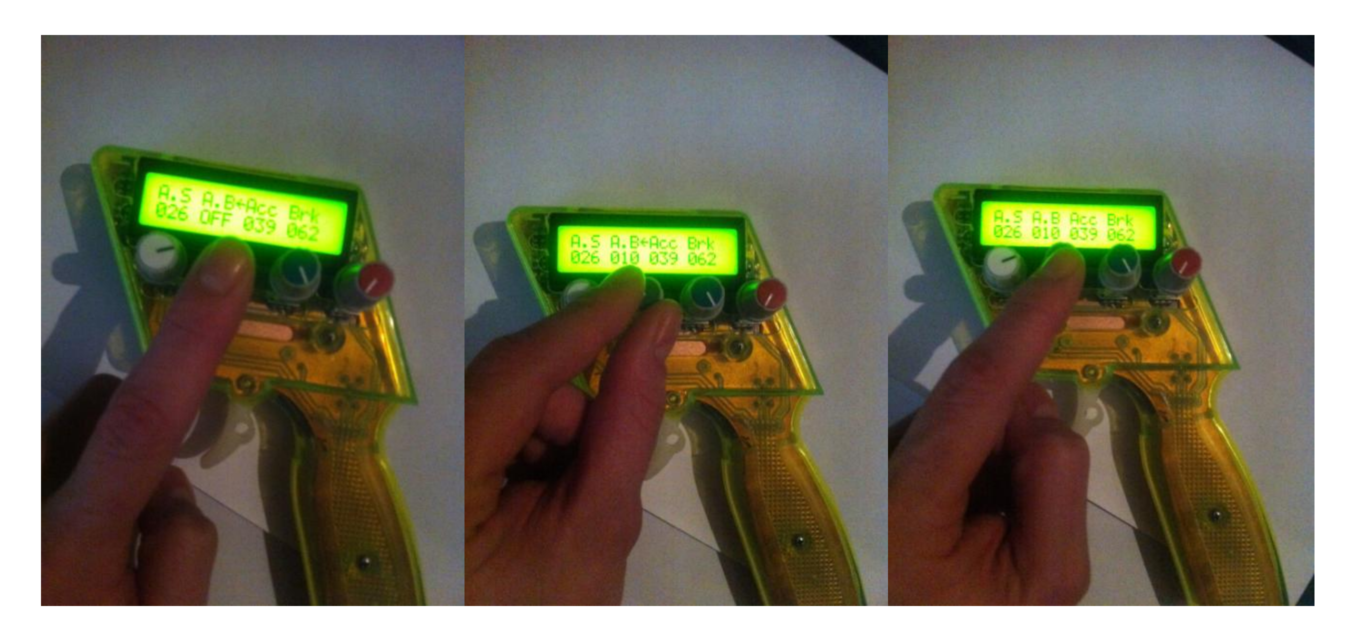


Figura 35. Ilustración de los pasos para la modificación de parámetros de configuración.

En el programa hay un menú que se accede pulsando de 2 a 4 segundos el 1º *encoder* (AS), a estas opciones se accederá volviendo a pulsar una sola vez el mismo *encoder.*
El menú está formado por cuatro opciones:

- **GUARDAR**: En la que el usuario puede guardar hasta 50 configuraciones distintas, con el mismo *encoder,* el usuario elegirá desde el 1 hasta el 50 girando a la derecha subiendo, y a la izquierda bajando el valor de la posición de memoria a guardar la configuración del usuario, pulsando otra vez el *encoder* empezará el proceso de guardado mostrándonos en el LCD un aviso de "ESPERE PORFAVOR", con una barra de tarea, cuando se haya acabado el proceso de guardado, saldrá directamente del menú.

- **CARGAR**: En esta opción, el usuario puede cargar las 50 configuraciones que han sido guardadas en memoria con anterioridad, con el mismo *encoder,* el usuario elegirá desde el 1 hasta el 50 girando a la derecha subiendo y a la izquierda bajando el valor de la posición de memoria a cargar la configuración del usuario, pulsando otra vez el *encoder* empezará el proceso de cargado mostrándonos en el LCD un aviso de "ESPERE PORFAVOR" con una barra de tarea, cuando se haya acabado el proceso de cargado saldrá directamente del menú. (**Figura 36**)



Figura 36. Ilustración de los pasos para guardado y cargado de los parámetros de configuración.

- **CALIBRADOR**: Esta opción sirve de ayuda tanto al fabricante, como al usuario, que le permite regular con exactitud la ganancia y el offset del circuito de acondicionamiento del sensor GMR, que sirve para ajustar los niveles de tensión, aptos para el conversor A/D, con el fin de reconocer la posición de todo el recorrido del gatillo. En el LCD nos mostrará dos números de cuatro dígitos cada uno, oscilando entre el 0000 y el 1023 (10 bits), el número de la derecha indica el valor del conversor A/D, que captura la posición del gatillo, este número debe ser 0000 cuando se encuentra en la posición de reposo, y 1023 cuando está en la posición máxima que se pueda accionar (si no fuera así, se debe ajustar estos valores con los dos potenciómetros que hay debajo del gatillo). El número de la izquierda indica el valor de ancho de pulso en que está trabajando el PWM de aceleración, dependiendo del valor del conversor A/D, este valor sirve para ayudar ajustar los parámetros de configuración del usuario con más exactitud, pudiendo ver numéricamente la respuesta del programa según la posición del gatillo. Pulsando de nuevo el *encoder,* el programa sale del menú (**Figura 37**).



Figura 37. Ilustración del calibrado del sensor GMR.

- **SALIR**: Esta opción sirve para salir del menú si se accede a él involuntariamente.

5.Bibliografía

- Electrónica de potencia : circuitos, dispositivos y aplicaciones
  Muhammad Harunur Rashid México etc. : Pearson Educación 2004 3ª ed.

- Electrónica de potencia : teoría y aplicaciones
  José Manuel Benavent García Antonio Abellán García; Emilio Figueres Amorós; Universidad Politécnica de Valencia Valencia : Universidad Politécnica de Valencia D.L. 1999

- El lenguaje de programación C : diseño e implementación de programas
  Félix García Carballeira Jesús Carretero Pérez; Javier Fernández Muñoz; Alejandro Calderón Mateos Madrid etc. : Pearson Educación 2002

- Fundamentos de programación en C
  Miguel Ángel Vega Rodríguez Juan Manuel Sánchez Pérez Cáceres : Universidad de Extremadura D.L. 2003

- Pic C : an introduction to programming the microchip PIC in C
  Nigel Gardner Brookfield : Bluebird Electronics 1998

- Embedded C programming and the microchip PIC
  Richard Barnett Larry O'Cull; Sarah Cox Clifton Park : Thomson/Delmar Learning cop. 2004

- Advanced PIC microcontroller projects in C : from USB to ZIGBEE with the 18F series
  Dogan Ibrahim Oxford : Newnes 2008

- Measurement, Instrumentation and Sensor Handbook,
  CRCNetBase 1999

- Sensores y Acondicionadores de señal.
  Ramón Pallas Areny. Ed. Marcombo 1994

- C. Reig, M.D. Cubells-Beltrán, D. Ramírez. ¿GMR based electrical current sensors¿ in ¿Giant Magnetoresistance: New Research¿. Nova Science Publishers (2009)

- C. Reig, M.D. Cubells-Beltrán, D. Ramírez. ¿Magnetic Field Sensors Based on Giant Magnetoresistance (GMR) Technology: Applications in Electrical Current Sensing¿. Sensors 9 (2009)

Páginas Web

- http://www.ronda.net/ocio/scalextric/historia.htm

- http://es.wikipedia.org/wiki/Magnetorresistencia_gigante

- http://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos

6.Anexos

Datasheets de los componentes más relevantes utilizados.

- Datasheet del LCD 2x16 caracteres.
- Datasheet de los encoders rotatorios.
- Datasheet de los Amplificadores Operacionales **LMV358SG-13**.
- Datasheet de transistor **NPN IRF540**.
- Datasheet del transistor **PNP IRF9540**.
- Datasheet del optoacoplador **TIL191**.
- Datasheet de los reguladores de tensión **78XX**.
- Datasheet del microcontrolador **PIC 18F4520**.

Esquemático y Layout de la placa impresa.

- Esquema de la etapa de control y de potencia.
- Layout de la placa impresa.
- Cara Top.
- Cara Bottom.

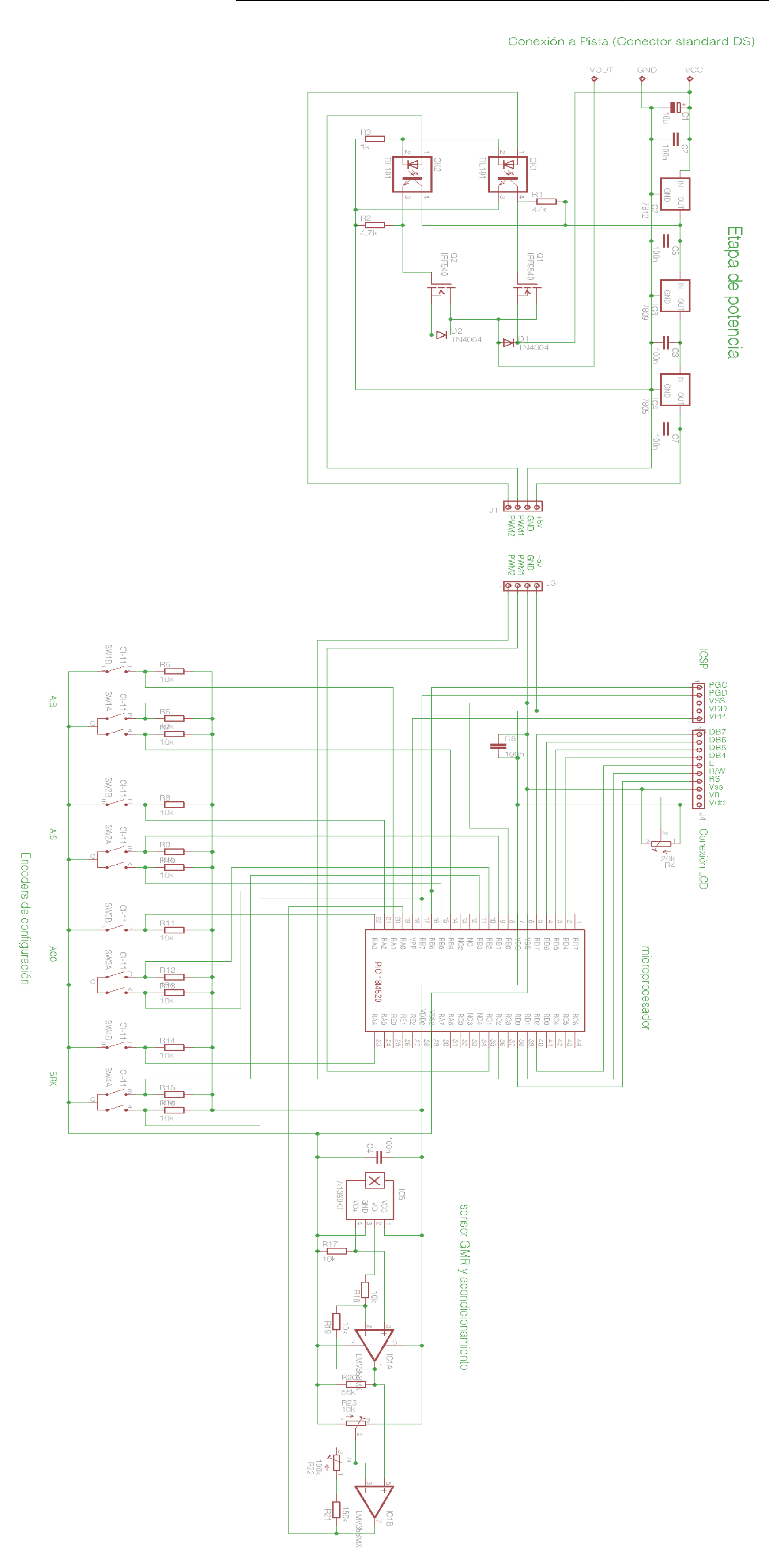
Archivos en código C para la programación del microcontrontrolador 18F4520.

- Programa escrito en código C para el PIC 18F4520.
- Librería lcd.h utilizada en el programa.

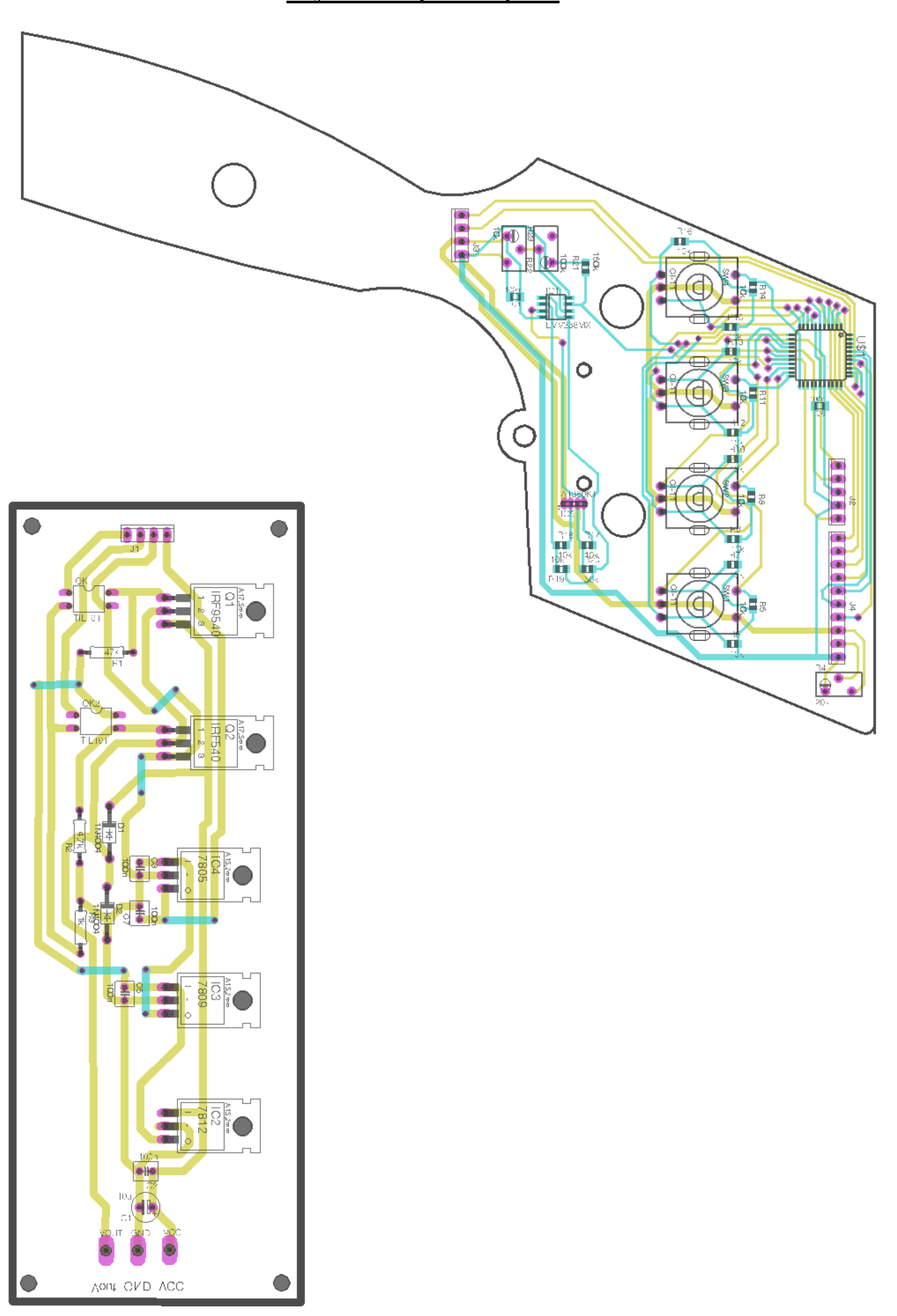
Presupuesto.

- Presupuesto para la producción de 100 unidades.

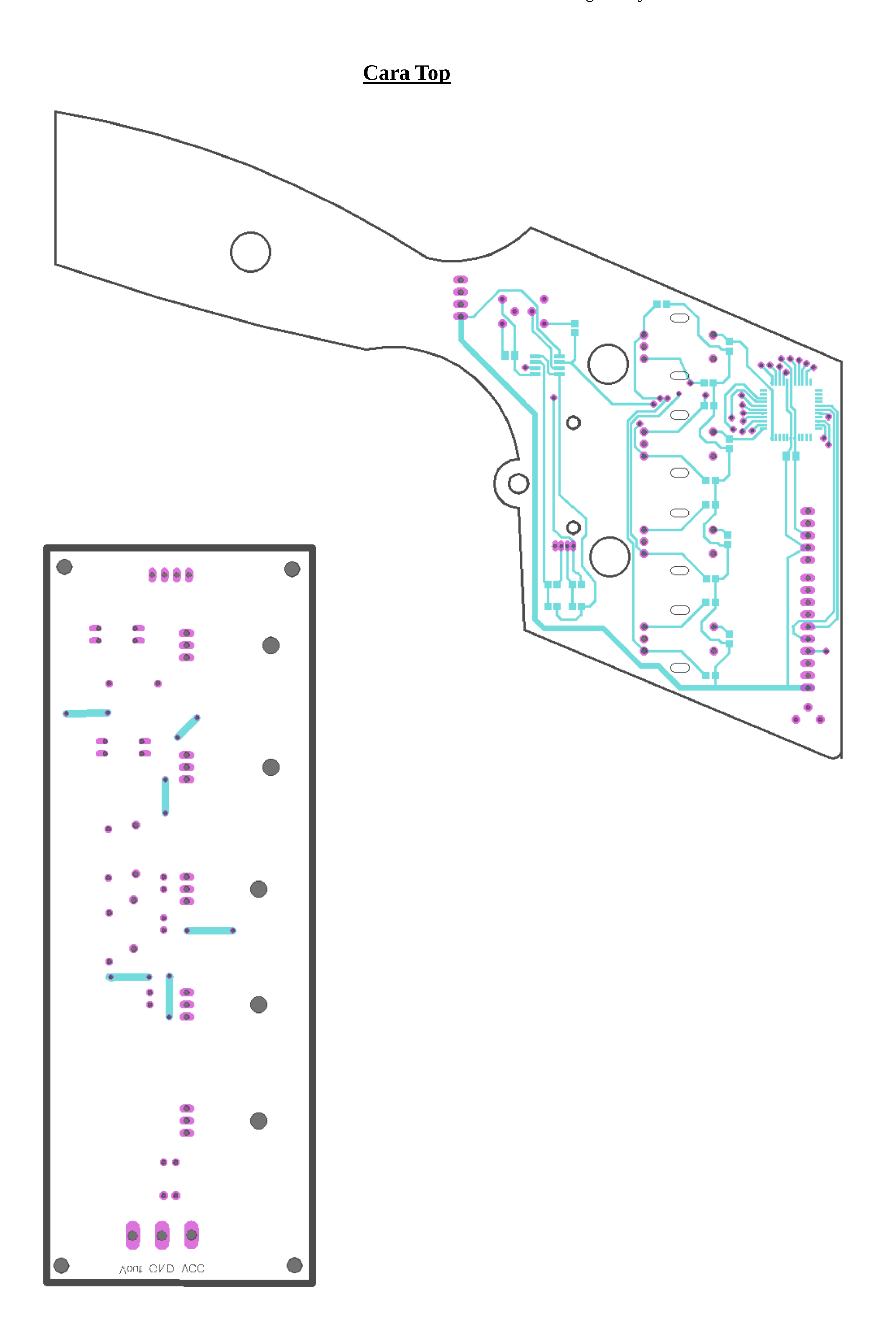
Esquema de la etapa de control y de potencia



Layout de la placa impresa



Cara Top



Cara Bottom

```
#include<pic18f4520.h>                    //añadimos la libreria del PIC a programar 18F4520
#include <htc.h>                          //añadimos la libreria de las funciones y nombres asignados a los puertos y registros del PIC
#include "C:\Projecte18f\lcd.h"           //añadimos la libreria de las funcionas que controlan el LCD con bus de datos a 4 bits por el puerto D



const unsigned char indice[]="A.S A.B Acc Brk ";        //Asignación de constantes tipo carácter para los indices y nevegación del menú

 const unsigned char menu[]="--MENU MEMORIA--";
 const unsigned char guarda[]="-----GUARDAR--->";
 const unsigned char guarda2[]="-----GUARDAR----";
 const unsigned char carga[]="<----CARGAR---->";
 const unsigned char carga2[]="-----CARGAR-----";
 const unsigned char espera[]="ESPERE PORFAVOR";
 const unsigned char elije[]="MEMORIA NUM: ";
 const unsigned char salir[]="<-----SALIR-----";
 const unsigned char calibrado[]="<--CALIBRADOR-->";
const unsigned char calibrado2[]="---CALIBRADOR---";


 const unsigned char puntos[]="||||";

const unsigned char apagado[]="OFF";


unsigned int y=0,x=0,z=0,f=0;                              //asignación de variables globales de programa

unsigned int datoad=0,valor_ccp=0,valor_ccp2=0,CCP=0,CCP2=0,datosen=0,ACCP=0,RCCP=0;


unsigned char AS=0,AS2=0,AB=0,ACC=0,BRK=0,BRK2=0;
unsigned char enc1=0,enc2=0,enc3=0,enc4=0,pos1=0,pos2=0,pos3=0,pos4=0,puntero=0,e=0,t=0,men=0,pmenu=0,pos=1,posm=0,s=0,m=0,tt=0,fas=0,filtro=0;

void envialcd( unsigned char pos,const unsigned char *dato);     //*dato es el puntero de direccion de las tablas


void guardar(unsigned char mempos);     // función para guardar datos de configuración en memoria EEPROM
void cargar(unsigned char mempos);      // función para cargar datos de configuración desde la memoria EEPROM

void main(void)     // inicio programa principal
{

    OSCTUNE =0x80;  // configuración reloj interno a 4MHz
    OSCCON =0x66;
    RCON =0x10;

    EECON1=0x00;    // configuracion lectura/escritura eeprom

    TRISB=0xFF;     //configuración puerto B de entrada
    TRISD=0x00;     //configuración puerto D de salida
```

```
TRISC=0x00;     //configuración puerto C de salida
TRISA=0x1F;     //configuración puerto A de entrada

EEPGD=0;        //configuración y habilitación de interrupción de lectura/escritura EEPROM
EEIE=1;
WREN=1;

INTCON=0xE8;        // Configuración y habilitación de interrupciones Puerto B y Timer 0
INTCON2=0x00;       // RPULLUP OFF-> 0x80 ONN

ADCON0=0x01;        //Configuración del convertidor A/D a un canal analógico del puerto A (Canal 0)
ADCON1=0x0E;
ADCON2=0x86;

T2CON=0x02;         // Configuración Timer 2 como PWM a 300MHz
PR2=255;
TMR2ON=1;

T0CON=0xC7;         //Configuración Timer 0 a 100Hz
TMR0L=61;

TMR1IE=1;           //Habilitación sw interrupción y configuración del Timer 1 a 100 MHz
T1CON=0x00;
TMR1H=0xD8;
TMR1L=0xF0;
TMR1ON=1;

ADIE=0;             //Deshabilitación de la interrupción del convertidor A/D
GO=1;               //Inicio de conversión

EEIF=0;             //Desactivación de todos los flags de interrupción
RBIF=0;
TMR0IF=0;
ADIF=0;

//////////////////   CARGA MEMORIA INICIO    ////////////////////

cargar(0x00);       //Carga datos de configuración en la posición de la memoria (Configuración actual)

//////////////////   INICIO LCD    ////////////////////

inicializar_lcd();          //Función para inicializar el LCD en lcd.h

envia_al_lcd(0,0x01);       //Funcion para borrar pantalla del LCD en lcd.h


do
{       TMR1ON=1;       // Activación del Timer 1 para el inicio de conversión A/D

        //////////////////   CONTROL LCD E INDICADOR/PUNTERO   ////////////////////
```

```
switch(puntero)     // este switch case envia al LCD un puntero para señalizar el parámetro a modificar
{                   // según el encoder que se ha pulsado
    case 1: {envia_al_lcd(0,0x83);envia_al_lcd(1,0x7F);break;}
    case 2: {envia_al_lcd(0,0x87);envia_al_lcd(1,0x7F);break;}
    case 3: {envia_al_lcd(0,0x8B);envia_al_lcd(1,0x7F);break;}
    case 4: {envia_al_lcd(0,0x8F);envia_al_lcd(1,0x7F);break;}
    default:{envialcd(0x80,indice);break;}  // envia la cabecera de indices de configuración

}



envia_al_lcd(0,0xC0);                   // Borra en el LCD el valor de configuración AS en la posición C0

if (AS==0){envialcd(0xC0,apagado);}     // Muestra en el LCD el valor de configuración AS en la posición C0
else
{
    envia_al_lcd(1,(AS/100)+0x30);  //sigue en la posicion donde se ha quedado si no asignamos posición
    envia_al_lcd(1,((AS%100)/10)+0x30);
    envia_al_lcd(1,(AS%10)+0x30);

}

envia_al_lcd(0,0xC4);                   // Borra en el LCD el valor de configuración AB en la posición C4

if (AB==0){envialcd(0xC4,apagado);}     // Muestra en el LCD el valor de configuración AB en la posición C4
else
{
    envia_al_lcd(1,(AB/100)+0x30);
    envia_al_lcd(1,((AB%100)/10)+0x30);
    envia_al_lcd(1,(AB%10)+0x30);
}

envia_al_lcd(0,0xC8);                   // Borra en el LCD el valor de configuración ACC en la posición C8

if (ACC==0){envialcd(0xC8,apagado);}    // Muestra en el LCD el valor de configuración ACC en la posición C8
else
{
envia_al_lcd(1,(ACC/100)+0x30);
envia_al_lcd(1,((ACC%100)/10)+0x30);
envia_al_lcd(1,(ACC%10)+0x30);
}

envia_al_lcd(0,0xCC);                   // Borra en el LCD el valor de configuración BRK en la posición CC

if (BRK==0){envialcd(0xCC,apagado);}    // Muestra en el LCD el valor de configuración ACC en la posición CC
else
{
envia_al_lcd(1,(BRK/100)+0x30);
```

```
envia_al_lcd(1,((BRK%100)/10)+0x30);
envia_al_lcd(1,(BRK%10)+0x30);
}

//////////////////   CONTROL ENCODERS (Activacion del puntero)   ////////////////////

if(RA1==0)      //  Detección de pulsación del encoder 1 y activación de variables de puntero de configuración
{   TMR0ON=1;

    for(z=0;z<=20000;z++);
    if(RA1==1)
    {
        switch(pos1)
        {
            case 0: {enc2=enc3=enc4=0;enc1=1;pos4=pos2=pos3=0;pos1=1;puntero=1;envialcd(0x80,indice);TMR0ON=0;t=0;break;}
            case 1: {enc1=0;pos1=0;puntero=0;TMR0ON=0;t=0;break;}

        }
    }
}


if(RA2==0)      //  Detección de pulsación del encoder 2 y activación de variables de puntero de configuración
{   for(z=0;z<=3500;z++);
    if(RA2==1)
    {
        switch(pos2)
        {
            case 0: {enc1=0;enc3=0;enc4=0;enc2=1;pos1=0;pos4=0;pos3=0;pos2=1;puntero=2;envialcd(0x80,indice);break;}
            case 1: {enc2=0;pos2=0;puntero=0;break;}
        }
    }
}

if(RA3==0)      //  Detección de pulsación del encoder 3 y activación de variables de puntero de configuración
{   for(z=0;z<=3500;z++);
    if(RA3==1)
    {
        switch(pos3)
        {
            case 0: {enc2=0;enc1=0;enc4=0;enc3=1;pos1=0;pos2=0;pos4=0;pos3=1;puntero=3;envialcd(0x80,indice);break;}
            case 1: {enc3=0;pos3=0;puntero=0;break;}
        }
    }
}

if(RA4==0)      //  Detección de pulsación del encoder 4 y activación de variables de puntero de configuración
{   for(z=0;z<=3500;z++);
    if(RA4==1)
    {
```

```
            switch(pos4)
            {
                case 0: {enc2=0;enc3=0;enc1=0;enc4=1;pos1=0;pos2=0;pos3=0;pos4=1;puntero=4;envialcd(0x80,indice);break;}
                case 1: {enc4=0;pos4=0;puntero=0;break;}
            }
        }
    }

//////////////////   MENU (GUARDAR/CARGAR/CALIBRADOR/SALIR)    ////////////////////


            if(tt==1)   // Condición activada por el timer si se ha pulsado continuamente el encoder 1 (RA1)
                {
                    envia_al_lcd(0,0x01);   // Borra LCD
                    envialcd(0x80,menu);    // Envía al LCD la cabezera menu

                    pmenu=1;                // Iniciación variables del control de menú
                    enc1=1;

                    while(RA1==0);          // Mientras no se vuelva a pulsar el encoder 1 (RA1) no saldrá del menú (while)

                    do
                    {
                        switch(men)         // Selección de las opciones del menú por el usuario con la variable men
                        {
                            case 0:{                            // Opción 0 guardar
                                        envialcd(0x80,menu);    // Envía al LCD la cabecera menú
                                        envialcd(0xC0,guarda);  // Envía al LCD en la sefunda linea la opción guardar
                                        if(RA1==0)              // Si se pulsa el pulsador del encoder 1 entrará en la opción guardar
                                        {   for(z=0;z<=35000;z++); // Bucle de espera
                                            if(RA1==1)          // Reconocimiento de pulsación para entrar en la opción guardar
                                            {   envia_al_lcd(0,0x01);   // Envía al LCD que borre pantalla
                                                do
                                                {   pmenu=2;
                                                    envialcd(0x80,guarda2); //Envía al LCD cabecera guarda2
                                                    envialcd(0xC0,elije);   //Envía al LCD el número de la posición de memora a guardar
                                                    envia_al_lcd(1,((pos%100)/10)+0x30);
                                                    envia_al_lcd(1,(pos%10)+0x30);
                                                    posm=pos*4;
                                                    s=1;
                                                }while(RA1==1); // Si se vuelve a pulsar el encoder 1 empieza el guardado de la configuración
                                                e=0;
                                                guardar(posm);  // Guarda la posición de memória el primer dato e configuración
                                                while(EEIF==1); //Espera al guardado en memória
                                                envia_al_lcd(0,0x01); //Borra pantalla de LCD
                                                envialcd(0x80,espera);  // Envía al LCD cabecera de espera
                                                envialcd(0xC0,puntos);  // Envía al LCD barras de estado de guardado
                                                for(z=0;z<=15000;z++);  //Ciclo de espera
                                                guardar(posm);          //Guardar en memória el segundo dato en la posición
                                                while(EEIF==1);         //de memória anterior indicada
```

```
                envialcd(0xC4,puntos);  //Envía al LCD otro tramo de barras de estado
                for(z=0;z<=15000;z++);
                guardar(posm);          //Guarda el segundo dato de configuración
                while(EEIF==1);
                envialcd(0xC8,puntos);  //Envía al LCD otro tramo de barras de estado
                for(z=0;z<=15000;z++);
                guardar(posm);          //Guarda el tercer dato de configuración
                while(EEIF==1);
                guardar(posm);          //Guarda el cuarto dato de configuración
                envialcd(0xCC,puntos);  //Envía al LCD el ultimo tramo de barras completando el guardado
                for(z=0;z<=15000;z++);
                envia_al_lcd(0,0x01);   //Borra el LCD
            }
        }

        pmenu=1;    //Inicializa la variable de seleccion de opciones del menú
        break;

    }

case 1:{                                //Opción 1 cargar
        envialcd(0x80,menu);        //Envía al LCD cabecera de menú
        envialcd(0xC0,carga);       //Envía ala segunda linea del LCD opción de carga
        if(RA1==0)                  //Si se pulsa el encoder 1 entrara a la opcion cargar
        {   for(z=0;z<=35000;z++);  //Bucle de espera
            if(RA1==1)              //Confirmación de la pulsación para entrar en opción cargar
            {   envia_al_lcd(0,0x01);  //Borra LCD
                do
                {   pmenu=2;                //Actualiza variable en la posición de opción
                    envialcd(0x80,carga2);  //Envía al lcd la cabecera de carga2
                    envialcd(0xC0,elije);   //Envía a la seguna linea la posición de memoria a cargar
                    envia_al_lcd(1,((pos%100)/10)+0x30);
                    envia_al_lcd(1,(pos%10)+0x30);
                    posm=pos*4;
                    s=1;
                }while(RA1==1);         // Si vuelve a pulsar activa la carga de datos de configuración
                cargar(posm);           //Función de carga de datos desde la memoria
                envia_al_lcd(0,0x01);   //Borra LCD
                envialcd(0x80,espera);  //Envía al LCD cabecera espera
            }
        }
        pmenu=1;                        //Inicialización variable de selección de opciones del menú
        break;
    }
case 2:{                                //Opción 2 calibrador
        envialcd(0x80,menu);            //Wnvía al LCD cabecera menú
        envialcd(0xC0,calibrado);       //Envía a la segunda linea del LCD la opcion calibrar
        if(RA1==0)                      //Si se pulsa el encoder 1 entraraa la opción calibrado
        {   for(z=0;z<=35000;z++);      //Bucle de espera
            if(RA1==1)                  //Confirmación de la pulsación para entrar en la opción calibrador
```

```
                        {   envia_al_lcd(0,0x01);   //Borra LCD
                            do
                            {   envialcd(0x80,calibrado2);              //Envía al LCD la cabecera calibrado2
                                envia_al_lcd(0,0xC0);                   //Borra la posición C0 del LCD
                                envia_al_lcd(1,(datoad/1000)+0x30);     //Envía a la posición C0 del LCD el valor datoad
                                envia_al_lcd(1,((datoad%1000)/100)+0x30);
                                envia_al_lcd(1,((datoad%100)/10)+0x30);
                                envia_al_lcd(1,(datoad%10)+0x30);
                                envia_al_lcd(0,0xC8);                   //Borra la posición C8 del LCD
                                envia_al_lcd(1,(valor_ccp/1000)+0x30);  //Envía a la posición C8 del LCD el valor
                                envia_al_lcd(1,((valor_ccp%1000)/100)+0x30); // del convertidor A/D
                                envia_al_lcd(1,((valor_ccp%100)/10)+0x30);
                                envia_al_lcd(1,(valor_ccp%10)+0x30);
                                s=1;
                            }while(RA1==1); // Si se viuelve a pulsar el encoder 1 sale del men´´u
                        }
                    }
                    pmenu=1;                //Inicialización de la variable de selección de opciones del menuú
                    break;
                }
            case 3:{                        //Opcioón 3 salir
                    envialcd(0x80,menu);    //Envía al LCD cabecera menu
                    envialcd(0xC0,salir);   //Envía a la segunda linea del LCD la opción salir
                    if(RA1==0)              //Si se pulsa el encoder 1 entra a la opción
                    {   for(z=0;z<=35000;z++);  //Bucle de espera

                        envia_al_lcd(0,0x01);   //Borra el LCD
                        s=1;

                    }
                    pmenu=1;        //Inicializa variable de selección de opciones del menú
                    break;
                }
            default:break;
        }

    }while(s==0);               //Mientas la variable s sea 0 estará dentro del menú
    s=0;                        //Inicialización de variable s a 0
    for(z=0;z<=20000;z++);      //bucle de espera
    envia_al_lcd(0,0x01);       //borra LCD
    enc1=0;                     //Inicialización de variables de control de menú, puntero y timer a 0
    pos1=0;
    puntero=0;
    pmenu=0;
    tt=0;
    t=0;
    TMR0ON=0;
}
```

```
        /////////////////   GUARDADO DE PARAMETROS AUTOMATICO EN MEMORIA   ////////////////////


        if(y==500)          //Bucle contador de vueltas de programa
        {                   //a cada 500 vueltas aproximadamente 1 segundo
            guardar(0x00);  //Función de guardado en la posición 00 de los parámetros de configuración
            y=0;            //Inicialización variable y del contador
        }
    y++;
    }
    while(1);               //repetición de programa principal


}


/////////////////   FUNCION LCD    ////////////////////


void envialcd(unsigned char pos, const unsigned char *dato) //Esta función recibe la posición de memoria pos y el dato tipo carácter para
                                                            // enviarla al LCD a través de la función envia_al_lcd de la librería lcd.h
{

    envia_al_lcd(0,pos);        //Envía al LCD la instrucción como posición de escritura

    do                          //Bucle para enviar todos los carácteres que contiene el dato
    {
        envia_al_lcd(1,*dato); //Envía el primer carácter de dato al LCD como dato
        dato++;
    }
    while(*dato!=0);            //Espera a que acabe de enviar al LCD todos los carácteres de dato

}


/////////////////   FUNCIONES GUARDAR/CARGAR     ////////////////////


void guardar(unsigned char mempos)                      // Esta función guarda los datos de configuración en la posición de memória mempos
{
        if(EEIF==0)                                     //Entra en la función si el flag de interrupción de la EEPROM esta a 0
        {
            switch(e)                                   //Elección según la variable e que controla que dato de configuración se está guardando
            {
                case 0:     {   EEADR=mempos;           //Opción 0 guarda en la posición de memória mempos el primer dato de configuración AS
                                RD=1;

                                if(EEDATA!=AS)          //Si el dato de configuración AS es diferente al que ya existe en memória lo guarda
```

```
                {
                    GIE=0;                  //Protocolo de guardado en memória EEPROM según el datasheet del fabricante
                    EEDATA=AS;
                    EEADR=mempos;
                    EECON2=0x55;
                    EECON2=0xAA;
                    WR=1;
                    GIE=1;
                }
                e=1;                        //Modifica la variable e para el guardado del segundo dato de configuración
                break;
            }
case 1:     {   EEADR=mempos|0x01;          //Opción 1 guarda en la posición de memória correlativa al de mempos
                RD=1;                       //el primer dato de configuración AB

                if(EEDATA!=AB)              //Si el dato de configuración AB es diferente al que ya existe en memória lo guarda
                {
                    GIE=0;                  //Protocolo de guardado en memória EEPROM según el datasheet del fabricante
                    EEDATA=AB;
                    EEADR=mempos|0x01;
                    EECON2=0x55;
                    EECON2=0xAA;
                    WR=1;
                    GIE=1;
                }
                e=2;                        //Modifica la variable e para el guardado del tercer dato de configuración
                break;
            }

case 2:     {   EEADR=mempos|0x02;          //Opción 2 guarda en la posición de memória correlativa al de mempos
                RD=1;                       //el primer dato de configuración ACC

                if(EEDATA!=ACC)             //Si el dato de configuración ACC es diferente al que ya existe en memória lo guarda
                {
                    GIE=0;                  //Protocolo de guardado en memória EEPROM según el datasheet del fabricante
                    EEDATA=ACC;
                    EEADR=mempos|0x02;
                    EECON2=0x55;
                    EECON2=0xAA;
                    WR=1;
                    GIE=1;
                }
                e=3;                        //Modifica la variable e para el guardado del cuarto dato de configuración
                break;
            }

case 3:     {   EEADR=mempos|0x03;          //Opción 3 guarda en la posición de memória correlativa al de mempos
                RD=1;                       //el primer dato de configuración BRK

                if(EEDATA!=BRK)             //Si el dato de configuración BRK es diferente al que ya existe en memória lo guarda
```

```
                                {
                                    GIE=0;              //Protocolo de guardado en memória EEPROM según el datasheet del fabricante
                                    EEDATA=BRK;
                                    EEADR=mempos|0x03;
                                    EECON2=0x55;
                                    EECON2=0xAA;
                                    WR=1;
                                    GIE=1;
                                }
                                e=0;                //Inicializa la variable e a 0 para volver a guardar los datos de configuración
                                break;
                            }
                default:    {break;}
            }
        }
}


void cargar(unsigned char mempos)       //Esta función carga los datos de configuración desde la posición de memória EEPROM  mempos
{

    EEADR=mempos;       //Lectura del primer dato de configuración AS
    RD=1;
    AS=EEDATA;

    EEADR=mempos+1;     //Lectura del segundo dato de configuración AB
    RD=1;
    AB=EEDATA;

    EEADR=mempos+2;     //Lectura del tercer dato de configuración ACC
    RD=1;
    ACC=EEDATA;

    EEADR=mempos+3;     //Lectura del cuarto dato de configuración BRK
    RD=1;
    BRK=EEDATA;


}

//////////////////   INTERUUPCIONES    ////////////////////



void interrupt mando (void)
{
    //////////////////   EPROM (EEINT)   ////////////////////

    if(EEIF){EEIF=0;} //Interrupción de lectura/escritura de la memória EEPROM pone el flag de interrupción a 0
```

```
//////////////////   TIMER1 (freno,accel,antspin)/CONTROL PWM   ////////////////////

if (TMR1IF==1)  //Interrupción del Timer 1 para la adquisición de datos del conversor operación y asignación para el PWM
{
    datoad=(ADRESH<<8)+ADRESL;                          //Lectura del dato convertido
    GO=1;                                               //Inicio nueva conversión


    if(datoad>=1)                                       //Si el dato que se ha adquirido del conversor A/D es mayor que uno
    {                                                   //entra al PWM de aceleración
            datosen=datoad;

                RC2=0;                                  //desactiva el PWM de freno


                if(ACC>=1)                              //Si la opcción de configuración ACC está activada (mayor que 0) entra
                {                                       //Incrementa el valor capturado dependiendo del valor de ACC
                    CCP=datosen+((1023/100)*(ACC/3));   //aumentando exponencialmente la curva de aceleración
                    if(CCP>=1023)                       //Suma en la tercera proporción del dato ACC el valor adquirido por datosen
                    {                                   //Si supera el máximo valor asigna el máximo valor posible
                        CCP=1023;
                    }
                }
                else
                {                                       //Si ACC está desactivado (a 0) no altera el valor de datosen
                    CCP=datosen;
                }



                if(AS>=1)                               //Si la opción de configuración AS está activada (mayor que cero) entra
                {                                       //Esta opción retarda el incremento del valor con repeticiones e
                                                        //incrementos pequeños del valor del dato adquirido
                    if(ACCP<CCP)                        //Si el dato anterior del CCP (ACCP) es menor que el actual CCP entra
                    {
                        if(AS<10)                       //Si la opción de configuración AS es menor que 10 entra
                        {
                            AS2=1;                      //Asignación del numero de incremento a sumar a AS2
                        }
                        else{AS2=AS/10;}                //Si AS es mayor que 10 asigna a AS2 el valor actual de AS partido por 10
                        if(fas==AS2)                    //Si fas es igual al valor de AS2 entra
                        {
                            fas=0;                      //Inicializa fas a 0
                            ACCP=ACCP+100;              //Incrementa el valor anterior en 100
                            if(ACCP>=CCP)               //Si el anterior valor es mas grande que el actual entra
                            {
                                ACCP=CCP;               //Asigna el valor actual al anterior
                            }
                        }
```

```
                fas++;                                  //Incrementa variable fas en 1

                CCP=ACCP;                               //Si fas no es igual que AS2 asigna el mismo valor anterior al actual
            }


        }

        ACCP=CCP;                                       //Asigna el valor actual al anterior

        valor_ccp=CCP;                                  //Asignación del valor actual al PWM de aceleración
        CCPR2L=valor_ccp>>2;
        CCP2CON=((valor_ccp<<4)&0x30)|0x0F;
        CCP1CON=0x00;


        datosen=0;                                      //Inicialización de variables a 0
        filtro=0;



}

    if (datoad==0)                                      //Si el valor del dato que se ha adquirido del conversor A/D es 0
    {                                                   //entra al PWM de freno
        datosen=0;                                      //Inicializa todas las variable del PWM de aceleración a 0
        valor_ccp=0;
        ACCP=0;
        CCP=0;
        RC1=0;                                          //Desactivación del PWM de aceleración
        if(BRK==0){RC2=0;CCP2CON=0x00;CCP1CON=0x00;}    //Si el dato de configuración BRK es 0 desactiva el PWM del freno

        else if(BRK==100){RC2=1;CCP2CON=0x00;}          //Si el dato de configuración es el máximo (100) el PWM del freno
        else                                            //trabajará con el máximo ancho de pulso
        {   BRK2=100-BRK;                               //Si el dato de configuración es difirente a los dos anteriores
                                                        //calcula el valor del ancho de pulso en proporción al dato de configuración BRK

            CCP2=(1023/100)*BRK2;

            valor_ccp2=CCP2;                            //Asignación del valor del ancho de pulso del PWM del freno
            CCPR1L=valor_ccp2>>2;
            CCP1CON=((valor_ccp2<<4)&0x30)|0x0F;
            CCP2CON=0x00;
        }
    }



TMR1H=0xD8;             //Reasignación del valor a temporizar
```

```
    TMR1L=0xF0;
    TMR1IF=0;               //Borrado de flag de interrupción del Timer 1

    TMR1ON=1;               //Inicialización de temporización del Timer 1
}


//////////////////   TIMER 0 (CONTROL TIEMPO PULSACION ENCODER PUNTERO)   ////////////////////

if(TMR0IF)                  //Interrupción del Timer 0 para el control del tiempo de pulsación del encoder 1
{t++;                       //Incremento de variable t en 1
    if(t==50)               //Si t es igual a 50 entra
    {
        if(RA1==0)          //Si el pulsador del encoder 1 esta pulsado entra
        {
            tt=1;           //Asignación de variable tt a 1 para acceder al menú

        }
        else{t=0;}          //Si el pulsador no se mantiene inicializa la variable t a 0
    }
    else
    {
        TMR0IF=0;           //Borrado de flag de interrupción Timer 0
        TMR0ON=1;           //Inicio de temporización del Timer 0
    }
}

//////////////////  INT PORT B (CONTROL ROTACION ENCODERS)     ////////////////////


if(RBIF) //Si se activa el flag de interrupción por cambio de nivel de las entradas RB4 RB5 RB6 o RB7 entra
{
    for(z=0;z<=40;z++);                 //bucle de espera

    if((RB4==1)&&(enc1==1))             //Si RB4 y enc1 son igual a 1 entra (enc1 es la variable que activa el pulsador del encoder 1 (RA1)
    {                                   //para la modificación del dato de configuración)
            if(RB0==0)                  //Si RB0 es igual a 0 entra (esta entrada es la segunda patilla del encoder que determina que
            {                           //el sentido de giro es a la izquierda)
                if(pmenu==0)            //Si pmenu es 0 entra (esta variable determina en que punto del menú se encuentra para que
                {                       //el mismo encoder modifique distintos datos)
                    if(AS>0){AS--;}     //Si el dato de la variable de configuración AS es mayor que 0 decrementa su valor en 1
                }
                if(pmenu==1)            //Si pmenu es igual a 1 entra para modificar el valor del dato men que controlará las opciones del menú
                {
                    if(men>0){men--;}   // Si el dato de la variable men es mayor que 0 decrementa el valor en 1
                }
                if(pmenu==2)            //Si pmenu es igual a 2 entra para modificar el valor del dato pos que controlará el número de
                {                       //la posición de memoria a guardar o cargar
                    if(pos>1){pos--;}   //Si el dato de la variable pos es mayor que 0 decrementa el valor en 1
                }
```

```
        }

}
if((RB4==0)&&(enc1==1))              //Si RB4 es igual a 0 y enc1 es igual a 1 entra
{
        if(RB0==0)                   //Si RB0 es igual 0 el sentido de giro del encoder es a la derecha
        {
            if(pmenu==0)             //Si pmenu es igual 0 modificara el dato de configuración AS
            {
                if(AS<100){AS++;}    //Si el dato de la variable de configuración AS es menor que 100 incrementa su valor en 1
            }
            if(pmenu==1)             //Si pmenu es igual a 1 entra para modificar el valor del dato men que controlará las opciones del menú
            {
                if(men<3){men++;}    //Si el dato de la variable men es menor que 3 incrementa su valor en 1
            }
            if(pmenu==2)             //Si pmenu es igual a 2 entra para modificar el valor del dato pos que controlará el número de
            {                        //la posición de memoria a guardar o cargar
                if(pos<50){pos++;}   //Si el dato de la variable pos es menor que 50 incrementa su valor en 1
            }
        }
}

if((RB5==1)&&(enc2==1))              //Si RB5 y enc2 son igual a 1 entra (enc2 es la variable que activa el pulsador del encoder 2 (RA2) para
{                                    //la modificación del dato de configuración)
    if(RB1==0)                       //Si RB1 es igual a 0 entra (esta entrada es la segunda patilla del encoder que determina que
    {                                //el sentido de giro es a la izquierda)
        if(AB>0){AB--;}              //Si el dato de la variable de configuración AB es mayor que 0 decrementa su valor en 1
    }
}
if((RB5==0)&&(enc2==1))              //Si RB5 es igual a 0 y enc2 es igual a 1 entra
{
    if(RB1==0)                       //Si RB1 es igual 0 el sentido de giro del encoder es a la derecha
    {
        if(AB<100){AB++;}            //Si el dato de la variable de configuración AB es menor que 100 incrementa su valor en 1
    }
}

if((RB6==1)&&(enc3==1))              //Si RB6 y enc3 son igual a 1 entra (enc3 es la variable que activa el pulsador del encoder 3 (RA3)
{                                    //para la modificación del dato de configuración)
    if(RB2==0)                       //Si RB2 es igual a 0 entra (esta entrada es la segunda patilla del encoder que determina que
    {                                //el sentido de giro es a la izquierda)
        if(ACC>0){ACC--;}            //Si el dato de la variable de configuración ACC es mayor que 0 decrementa su valor en 1
    }
}
if((RB6==0)&&(enc3==1))              //Si RB6 es igual a 0 y enc3 es igual a 1 entra
{
    if(RB2==0)                       //Si RB2 es igual 0 el sentido de giro del encoder es a la derecha
    {
        if(ACC<100){ACC++;}          //Si el dato de la variable de configuración ACC es menor que 100 incrementa su valor en 1
    }
```

```
        }

        if((RB7==1)&&(enc4==1))             //Si RB7 y enc4 son igual a 1 entra (enc3 es la variable que activa el pulsador del encoder 4 (RA4)
        {                                   //para la modificación del dato de configuración)
            if(RB3==0)                      //Si RB3 es igual a 0 entra (esta entrada es la segunda patilla del encoder que determina que
            {                               //el sentido de giro es a la izquierda)
                if(BRK>0){BRK--;}           //Si el dato de la variable de configuración BRK es mayor que 0 decrementa su valor en 1
            }
        }
        if((RB7==0)&&(enc4==1))             //Si RB7 es igual a 0 y enc4 es igual a 1 entra
        {
            if(RB3==0)                      //Si RB7 es igual 0 el sentido de giro del encoder es a la derecha
            {
                if(BRK<100){BRK++;}         //Si el dato de la variable de configuración BRK es menor que 100 incrementa su valor en 1
            }
        }

    RBIF=0;                                 //Inicializacion flag de interrupción port b a 0
    }

}                                           //FIN DEL PROGRAMA
```

Presupuesto de los componente para la elaboración de una unidad

COMPONENTES ELECTRÓNICOS	PRECIO UD.		UNIDADES		PRECIO
BOURNS - PEC11-4230F-S0024 - INCREMENTAL ENCODER	0,79 €		4		3,16 €
ISOCOM - TIL191 - OPTOACOPLADOR	0,19 €		2		0,38 €
STMICROELECTRONICS - LMV358ID - AMPLIFICADOR OPERACIONAL, R/R DUAL	0,44 €		1		0,44 €
SENSOR GMR EXPERIMENTAL - SS501A	2,00 €		1		2,00 €
MICROCHIP - PIC18F4520-I/PT - CI, MCU FLASH 8 BITS, 18F4520, TQFP44	3,46 €		1		3,46 €
MIDAS - MC21605DA6W-SPTLY - LCD, 2 x 16, STN, YLW/GREEN B/L, 5MM	6,89 €		1		6,89 €
INTERNATIONAL RECTIFIER - IRF540ZPBF - MOSFET, N, 100V, 36A, TO-220	1,00 €		1		1,00 €
INTERNATIONAL RECTIFIER - IRF9540NPBF - MOSFET, P, -100V, -23A, TO-220	1,08 €		1		1,08 €
STMICROELECTRONICS - L7805ABP - CI, REG LDO, 5V, 1A, TO220	0,95 €		1		0,95 €
TAIWAN SEMICONDUCTOR - TS7809CZ - CI, REG. DE TENSIÓN 9V, TO-220-3	0,18 €		1		0,18 €
FAIRCHILD SEMICONDUCTOR - LM7812CT. - REG. DE TENSIÓN, 14.5V, TO-220	0,42 €		1		0,42 €
BOURNS - CR0805-FX-1002ELF - RESISTENCIA, 0805, 10KR, 1%, 125mW	0,02 €		15		0,30 €
MULTICOMP - MCTC0525B1503T5E - RESISTENCIA, 150K, 0805 0.1% 0.1W	0,04 €		1		0,04 €
TE CONNECTIVITY / NEOHM - CPF0805B56KE1 - RESISTENCIA, 0805, 56K	0,35 €		1		0,35 €
MULTICOMP - MCRE000037 - RESISTENCIA, PELÍCULA DE CARBÓN, 0.125W, 1K	0,01 €		1		0,01 €
MULTICOMP - MCRE000045 - RESISTENCIA, PELÍCULA DE CARBÓN, 125mW, 4K7	0,01 €		1		0,01 €
MULTICOMP - MCRE000057 - RESISTENCIA, PELÍCULA DE CARBÓN, 125mW, 47K	0,01 €		1		0,01 €
VISHAY SFERNICE - T93XB203KT20 - AJUSTABLE, 22 VUELTAS 20K	0,73 €		1		0,73 €
BI TECHNOLOGIES/TT ELECTRONICS - 67WR10KLF - TRIMMER 500MW	0,81 €		1		0,81 €
VISHAY SFERNICE - T93YA104KT20 - AJUSTABLE, 22 VUELTAS 100K	0,77 €		1		0,77 €
FAIRCHILD SEMICONDUCTOR - 1N4007 - DIODE, STANDARD, 1A, 1000V	0,15 €		2		0,30 €
ILLINOIS CAPACITOR - 107CKH025M - CAPACITOR ALUM ELEC 100UF, 25V	0,15 €		2		0,30 €
VISHAY SPRAGUE - 489D106X0016C6VE3 - CONDENSADOR, TANTALIO, 16V, 10 uF	0,67 €		1		0,67 €
VISHAY SPRAGUE - 1C10C0G101J100B - CONDENSADOR, 100PF, 100V, C0G	0,13 €		1		0,13 €
AVX - 08051C104K4T2A - CONDENSADOR, 0805, X7R, 100V, 100 nF	0,09 €		1		0,09 €
COMPONENTES DE ENSAMBLAJE					
MULTICOMP - CP-AA15-3/CA - TAPA, AZUL	0,06 €		1		0,06 €
MULTICOMP - CP-AA15-1/CA - TAPA, ROJO	0,06 €		1		0,06 €
MULTICOMP - CP-AA15-7/CA - TAPA, BLANCO	0,06 €		1		0,06 €
MULTICOMP - CP-AA15-5/CA - TAPA, GRIS	0,06 €		1		0,06 €
MULTICOMP - CP-AA15-T18/B5 - MANDO, 15MM, GRIS	0,12 €		4		0,48 €
MULTICOMP - BM11W - CAJA, ABS, BLANCO, 30 x 143 x 82 mm	3,60 €		1		3,60 €
GATILLO NYLON - PM2018	6,00 €		1		6,00 €
CARCASA MANDO PROFESSOR MOTOR - PM2017	12,00 €		1		12,00 €
CONECTOR COMPACTO 3 BANANAS DS - DS0010	4,00 €		1		4,00 €
ECLIPSE MAGNETICS - N807 - NEODYMIUM DISC MAGNETS 6MM X 3MM	0,91 €		1		0,91 €
BRAND REX - GPU-HF1-RLX - GUÍA, CAT5 UTP, LSZH, POR M	0,95 €		1		0,95 €
PLACA IMPRESA FABRICADA POR EUROCIRCUITS (BE.EUROCIRCUITS.COM)	12,00 €		1		12,00 €

* Observaciones: estos son precios de venta para una venta superior a cien unidades por componente.

Total unitario 64,66 €

Coste de producción 100 unidades **64,66 € X 100 = 6.466,00 €**

Presupuesto realizado el 05/02/2013